

AD-A153 988

SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES VOLUME 1

1/2

FINAL REPORT(U) BOEING AEROSPACE CO SEATTLE WA

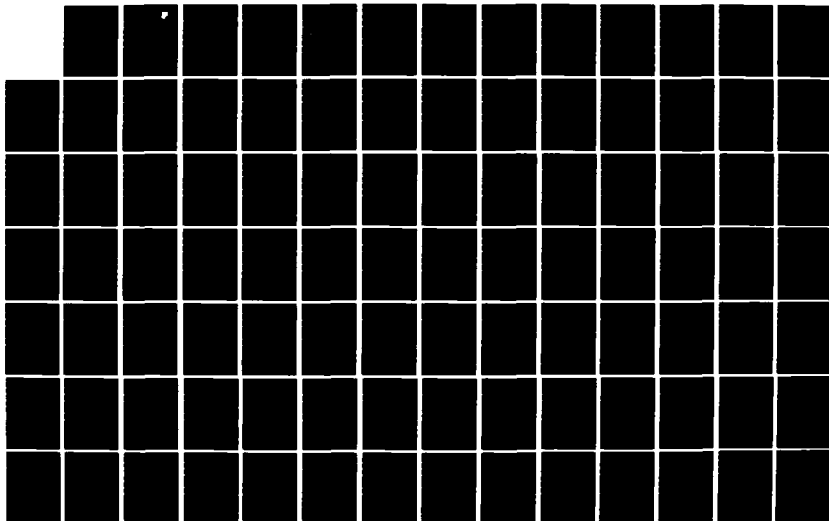
T P BOWEN ET AL. FEB 85 D182-11678-1

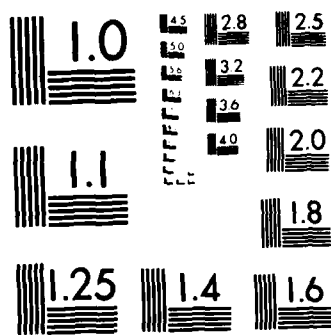
UNCLASSIFIED

RADC-TR-85-37-VOL-1 F30602-82-C-0137

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(2)

RADC-TR-85-37, Vol I (of three)
Final Technical Report
February 1985

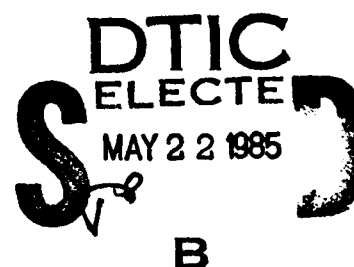


AD-A153 988

***SPECIFICATION OF SOFTWARE
QUALITY ATTRIBUTES***

Boeing **Aerospace Company**

Thomas P. Bowen, Gary B. Wigle and Jay T. Tsai



APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC FILE COPY

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

85 4 23 207

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

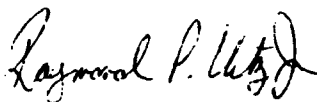
RADC-TR-85-37, Volume I (of three) has been reviewed and is approved for publication.

APPROVED:



ROGER B. PANARA
Project Engineer

APPROVED:



RAYMOND P. URTZ, JR.
Technical Director
Command & Control Division

FOR THE COMMANDER:



DONALD A. BRANTINGHAM
Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COEE) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS N/A		
2a. SECURITY CLASSIFICATION AUTHORITY N/A			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) D182-11678-1			5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-85-37, Vol I (of three)		
6a. NAME OF PERFORMING ORGANIZATION Boeing Aerospace Company		6b. OFFICE SYMBOL (If applicable)		7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (COEE)	
6c. ADDRESS (City, State and ZIP Code) P.O. Box 3999 Seattle WA 98124			7b. ADDRESS (City, State and ZIP Code) Griffiss AFB NY 13441-5700		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Rome Air Development Center		8b. OFFICE SYMBOL (If applicable) COEE		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F30602-82-C-0137	
8c. ADDRESS (City, State and ZIP Code) Griffiss AFB NY 13441-5700			10. SOURCE OF FUNDING NOS.		
			PROGRAM ELEMENT NO. 63728F	PROJECT NO. 2527	TASK NO. 03
			WORK UNIT NO. 05		
11. TITLE (Include Security Classification) SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES					
12. PERSONAL AUTHOR(S) Thomas P. Bowen, Gary B. Wigle, Jay T. Tsai					
13a. TYPE OF REPORT Final		13b. TIME COVERED FROM Aug 82 to Oct 84		14. DATE OF REPORT (Yr., Mo., Day) February 1985	
15. PAGE COUNT 122					
16. SUPPLEMENTARY NOTATION N/A					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB. GR.	Software Quality		
09	02		Software Quality Metrics		
19. ABSTRACT (Continue on reverse if necessary and identify by block number)					
<p>Volume I (of three) describes the results and presents recommendations for integrating the RADC developed software quality metrics technology into the Air Force software acquisition management process and for changing Air Force acquisition documentation. In addition, changes to the baseline software quality framework are presented and features of a proposed specification methodology are summarized. Terminology and life cycle phases are consistent with the December 1983 draft of the DOD-STD-SDS, Defense System Software Development.</p> <p>Volume II (of three) describes how the software acquisition manager specifies software quality requirements, consistent with needs. Factor interrelationships, tradeoff among factor quality levels in terms of relative costs and an example for a command and control application are described. Procedures for assessing compliance with the specified requirements based on an analysis of data collected using procedures described in volume III are included.</p>					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input checked="" type="checkbox"/> DTIC USERS <input type="checkbox"/>			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL Roger B. Panara			22b. TELEPHONE NUMBER (Include Area Code) (315) 330-4654		22c. OFFICE SYMBOL RADC (COEE)

DD FORM 1473, 83 APR

EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

SECURITY CLASSIFICATION OF THIS PAGE

DTIC
ELECTE
MAY 22 1985

[illegible]

SECURITY CLASSIFICATION OF THIS PAGE

PREFACE

This document is the first of three volumes of the Final Technical Report (CDRL A004) for the Specification of Software Quality Attributes contract, F30602-82-C-0137. Contract work was performed by Boeing Aerospace Company (BAC) for Rome Air Development Center (RADC) to provide methods, techniques, and guidance to Air Force software acquisition managers who specify the requirements for software quality.

The purpose of this contract was to (1) consolidate results of previous RADC contracts dealing with software quality measurement, (2) enhance the software quality framework, and (3) develop a methodology to enable a software acquisition manager to determine and specify software quality factor requirements. We developed the methodology and framework elements to focus on an Air Force software acquisition manager specifying quality requirements for embedded software that is part of a command and control application. This methodology and most of the framework elements are generally useful for other applications and different environments.

The Final Technical Report consists of three volumes:

- a. Volume I, Specification of Software Quality Attributes—Final Report.
- b. Volume II, Specification of Software Quality Attributes—Software Quality Specification Guidebook.
- c. Volume III, Specification of Software Quality Attributes—Software Quality Evaluation Guidebook.

Volume I describes the results of research efforts conducted under this contract, including recommendations for integrating quality metrics technology into the Air Force software acquisition management process, recommended changes to Air Force software acquisition documentation, and summaries of software quality framework changes and specification methodology features.

Volumes II and III describe the methodology for using the quality metrics technology and include an overview of the software acquisition process using this technology and the quality framework. Volume II describes methods for specifying software quality requirements and addresses the needs of the software acquisition manager. Volume III

describes methods for evaluating achieved quality levels of software products and addresses the needs of data collection and analysis personnel.

Volume II also describes procedures and techniques for specifying software quality requirements in terms of quality factors and criteria. Factor interrelationships, relative costs to develop high quality levels, and an example for a command and control application are also described. Procedures for assessing compliance with specified requirements are included.

Volume III also describes procedures and techniques for evaluating achieved quality levels of software products. Worksheets for collecting metric data by software life-cycle phase and scoresheets for scoring each factor are provided in the appendixes. Detailed metric questions on worksheets are nearly identical to questions in the Software Evaluation Reports proposed as part of the Software Technology for Adaptable Reliable Systems (STARS) Measurement data item descriptions (DID).

Terminology and life-cycle phases used in the guidebooks are consistent with the December 1983 draft of the Department of Defense software development standard (DOD-STD-SDS) (e.g., the term computer software configuration item (CSCI) is used rather than computer program configuration item (CPCI)).

CONTENTS

	<u>Page</u>
1.0 EXECUTIVE SUMMARY	1-1
1.1 Overview of Contract Results	1-1
1.2 Objectives	1-2
1.3 Background	1-3
1.4 Technical Approach	1-3
1.5 Task Approach and Accomplishments	1-5
1.6 Conclusions	1-11
2.0 ROLE OF QUALITY METRICS IN THE SOFTWARE ACQUISITION PROCESS	2-1
2.1 Software Acquisition Process	2-3
2.1.1 System Acquisition Life Cycle	2-3
2.1.2 Software Development Cycle	2-5
2.1.3 Life-Cycle Relationships	2-7
2.1.4 Software Acquisition Management	2-8
2.1.5 Verification and Validation	2-9
2.1.6 Quality Assurance	2-11
2.2 Quality Metrics	2-13
2.2.1 Framework	2-15
2.2.2 Quality Specification	2-15
2.2.3 Quality Monitoring	2-22
2.3 Software Acquisition Using Quality Metrics	2-23
2.4 Implementing Quality Metrics	2-31
2.4.1 Near-Term Implementation	2-31
2.4.2 Long-Term Implementation	2-32
2.5 Potential Benefits and Problems	2-35
2.5.1 Benefits	2-35
2.5.2 Problems	2-36
3.0 QUALITY METRICS FRAMEWORK	3-1
3.1 Software Quality Factors	3-3

	<u>Page</u>
3.1.1 Factor Definitions and Rating Formulas	3-5
3.1.1.1 Description	3-5
3.1.1.2 Changes	3-11
3.1.2 Quality Factor Interrelationships	3-13
3.1.3 Quality Factor Relationships to Life-Cycle Phases	3-13
3.2 Software Quality Criteria	3-13
3.2.1 Description	3-13
3.2.2 Changes	3-15
3.3 Software Quality Metrics	3-15
3.3.1 Description	3-15
3.3.2 Changes	3-18
3.4 Metric Worksheets	3-21
3.4.1 Description	3-21
3.4.2 Changes	3-23
3.5 Factor Scoresheets	3-26
3.5.1 Description	3-26
3.5.2 Changes	3-27
3.6 References	3-27
 4.0 QUALITY METRICS METHODOLOGY	 4-1
4.1 Overview	4-1
4.2 Features	4-5
 5.0 VALIDATION PLAN	 5-1
 6.0 RECOMMENDED REVISIONS	 6-1
6.1 Review and Recommendation Process	6-1
6.2 Review Analysis	6-3
6.3 Detailed Recommended Changes	6-5
6.3.1 DOD-STD-SDS	6-5
6.3.2 DI-S-X101 System/Segment Specification	6-7
6.3.3 DI-R-X105 Software Quality Assurance Plan	6-7
6.3.4 DI-E-X107 Software Requirements Specification	6-7

	<u>Page</u>
6.3.5 DOD-STD-SQS	6-8
6.3.6 MIL-STD-1521B	6-8
6.3.7 AFR 800-14	6-9
6.3.8 Guidebooks	6-10
Appendix A—Metric Worksheets	A-1
Appendix B—Factor Scoresheets	B-1
Appendix C—Software Quality Evaluation Report	C-1

FIGURES

	<u>Page</u>
1.3-1 Software Quality Model	1-4
1.5-1 Task Flow Diagram	1-6
1.5-2 Quality Metrics Technology—Life-Cycle Model	1-8
1.5-3 Software Quality Measurement Methodology	1-10
2.1-1 System Acquisition Life-Cycle Phases and Decision Points	2-2
2.1-2 Software Development Cycle	2-4
2.1-3 Life-Cycle Relationship between the System and the Operational Software	2-6
2.1-4 Relationship of Software Development and V&V	2-10
2.1-5 Software QA Function	2-12
2.2-1 Performance Factor Attributes	2-17
2.2-2 Design Factor Attributes	2-18
2.2-3 Adaptation Factor Attributes	2-19
2.3-1 Software Acquisition Quality Metrics Functions	2-24
2.3-2 Air Force Acquisition Relationships Involved in Quality Metrics Functions	2-25
2.3-3 Recommended Responsibilities and Relationships for the QM Specification Function	2-26
2.3-4 Recommended Responsibilities and Relationships for the QM Monitoring Function	2-30
2.4-1 Relationship between Product Divisions and DACS	2-34
3.1-1 Rating Estimation and Rating Assessment Windows	3-6
4.0-1 Software Quality Specification and Evaluation Process	4-2
4.0-2 Flow of Software Quality Requirements	4-4

TABLES

	<u>Page</u>
2.2-1 Quality Concerns	2-16
2.2-2 Software Quality Factor Interrelationships	2-20
2.3-1 Organizational Evaluation	2-28
3.1-1 Software Quality Factor Definitions and Rating Formulas	3-4
3.1-2 Quality Factor Ratings	3-7
3.2-1 Software Quality Factors and Criteria	3-12
3.2-2 Quality Criteria Definitions	3-14
3.3-1 Quality Metrics Summary	3-16
3.4-1 Metric Worksheet/Life-Cycle Correlation	3-22
3.4-2 Software Development Products	3-24
6.1-1 Candidate Documents	6-2
6.3-1 Documentation Recommended for Revision	6-6

GLOSSARY

AFCMD	Air Force Contracts Management Division
AFCL	Air Force Logistics Command
AFPRO	Air Force Plant Representative Office
AFSC	Air Force System Command
AMT	Automated Measurement Tool
APSE	Ada programming support environment
ASD	Aeronautical Systems Division
BAC	Boeing Aerospace Company
CDR	critical design review
CPCI	computer program configuration item
CSC	computer software component
CSCI	computer software configuration item
DACS	Data and Analysis Center for Software
DAE	Defense Acquisition Executive
DID	data item description
DOD	Department of Defense
DOD-STD-SDS	Department of Defense software development standard
DOD-STD-SQS	Department of Defense software quality standard
DRC	Dynamics Research Corporation
ESD	Electronic Systems Division
FCA	functional configuration audit
FSD	full-scale development
HOL	high order language
I/O	input/output
IV&V	independent validation and verification
OPR	Office of Primary Responsibility
PCA	physical configuration audit
PDR	preliminary design review
QA	quality assurance
QM	quality metrics
RADC	Rome Air Development Center
SD	Space Division

SDR	system design review
SPO	System Program Office
SSR	software specification review
STARS	Software Technology for Adaptable Reliable Systems
S/W	software
TRR	test readiness review
V&V	verification and validation

1.0 EXECUTIVE SUMMARY

Work described in this document was performed by Boeing Aerospace Company (BAC) for Rome Air Development Center (RADC), Griffiss Air Force Base, New York, under the Specification of Software Quality Attributes contract, F30602-82-C-0137. In this section, contract results are summarized, contract objectives are outlined, background information is provided, the technical approach to achieving objectives and major accomplishments are summarized, and several conclusions are explored.

1.1 OVERVIEW OF CONTRACT RESULTS

BAC extended and enhanced RADC quality metrics (QM) technology into a form usable by an Air Force software acquisition manager. RADC QM technology work began in 1976 as an effort to extend Quality Assurance (QA) beyond an administrative checklist type of activity to include quantitative considerations of software quality. This was achieved through the two processes of QM technology: (1) specifying software quality requirements in terms of quality factors (e.g. reliability, maintainability) and (2) periodically evaluating achieved quality levels throughout software development. These two processes are complementary in that periodic measurements enable a comparison with specified requirements.

The Air Force acquisition manager specifies software quality requirements concurrently with technical performance and design requirements, after consultation with user and logistics personnel. Procedural steps in the QM methodology guide the acquisition manager in developing software quality requirements and in making trade-off decisions that include quality considerations. Trade-off techniques include relative cost considerations for factor qualities over the system life cycle, the relative importance of factors and factor attributes, and the feasibility of achieving quality levels for specific factor considerations.

Points in the software development cycle at which quality levels are measured and reported coincide with review and audit points specified in the proposed Department of Defense standard for software development (DOD-STD-SDS). Source materials for data collection are data item descriptions (DID) identified in DOD-STD-SDS.

Measurements are, for the most part, consistent with those being specified in the Software Evaluation Report DIDs for Software Technology for Adaptable Reliable Systems (STARS). A system for reporting measurement data and results provides timely feedback to the acquisition manager and enables decision making and corrective action early in the development cycle for issues that may adversely affect cost and schedule.

QM technology has matured beyond the research stage and is ready for the test of use by acquisition managers. Procedural steps in the methodology have been documented in two guidebooks—one for specification and one for evaluation.

1.2 OBJECTIVES

Primary objectives for this contract were (1) to develop a methodology to enable a software acquisition manager to determine and specify software quality factor requirements and (2) to enhance the software quality framework. The methodology for determining and specifying quality factor requirements is focused on an Air Force acquisition manager procuring embedded software that is part of a command and control application. The software quality framework consists of the software quality factors, attributes of those factors (criteria and metrics), and the mechanisms (e.g., forms, tables, worksheets, and scoresheets) provided to enable specifying quality factor requirements and evaluating achieved quality levels.

Another objective for this contract was to prepare the QM technology for possible use by Air Force product divisions in acquiring new products. This involved defining an approach to integrating this technology into the Air Force software acquisition management process and ensuring that the technology elements were consistent with current Department of Defense (DOD) concepts. For example, life-cycle phases and terminology used in the quality metrics technology should be consistent with DODD 5000.1 and DOD-STD-SDS.

Another objective for this contract was to prepare a plan for validating the specification methodology to ensure its usability within software acquisition management.

1.3 BACKGROUND

There has been a recent increased awareness of critical problems encountered in developing large-scale systems involving software. These problems include cost and schedule overruns, high cost sensitivity to changes in requirements, poor performance of delivered systems, high system-maintenance costs, and lack of reusability.

The government (DOD in particular), as a customer for large-scale system developments, has sponsored efforts to address these problems. For example, development of Ada programming language, Ada programming support environments (APSE), proposed standards for software development (DOD-STD-SDS) and software quality (DOD-STD-SQS), the STARS program, the proposed STARS measurement DIDs, and various development aids and tools. These all provide partial solutions.

Since 1976, RADC has pursued a program intended to achieve better control of software quality. Through a series of related contracts, this program has sought to identify key software quality issues and to provide a valid methodology for specifying software quality requirements for software developed as part of major Air Force weapon systems and for measuring achieved quality levels of software products released incrementally during the software life cycle. A quality model was established (see Fig. 1.3-1) in which a hierarchical relationship exists between a user-oriented quality factor at the top level and software-oriented attributes at the second and third levels (criteria and metrics, respectively). Software quality is measured and predicted by the presence, absence, or degree of identifiable software characteristics.

The Final Technical Report for this contract consisting of the Final Report (Vol. I), the Software Quality Specification Guidebook (Vol. II), and the Software Quality Evaluation Guidebook (Vol. III) represents the most recent results of the RADC software quality program and incorporates pertinent results from previous contracts.

1.4 TECHNICAL APPROACH

The general approach used in accomplishing the objectives described in Section 1.2 was to first develop an approach for integrating QM technology into the Air Force software acquisition management process. This resulted in definitions of life-cycle

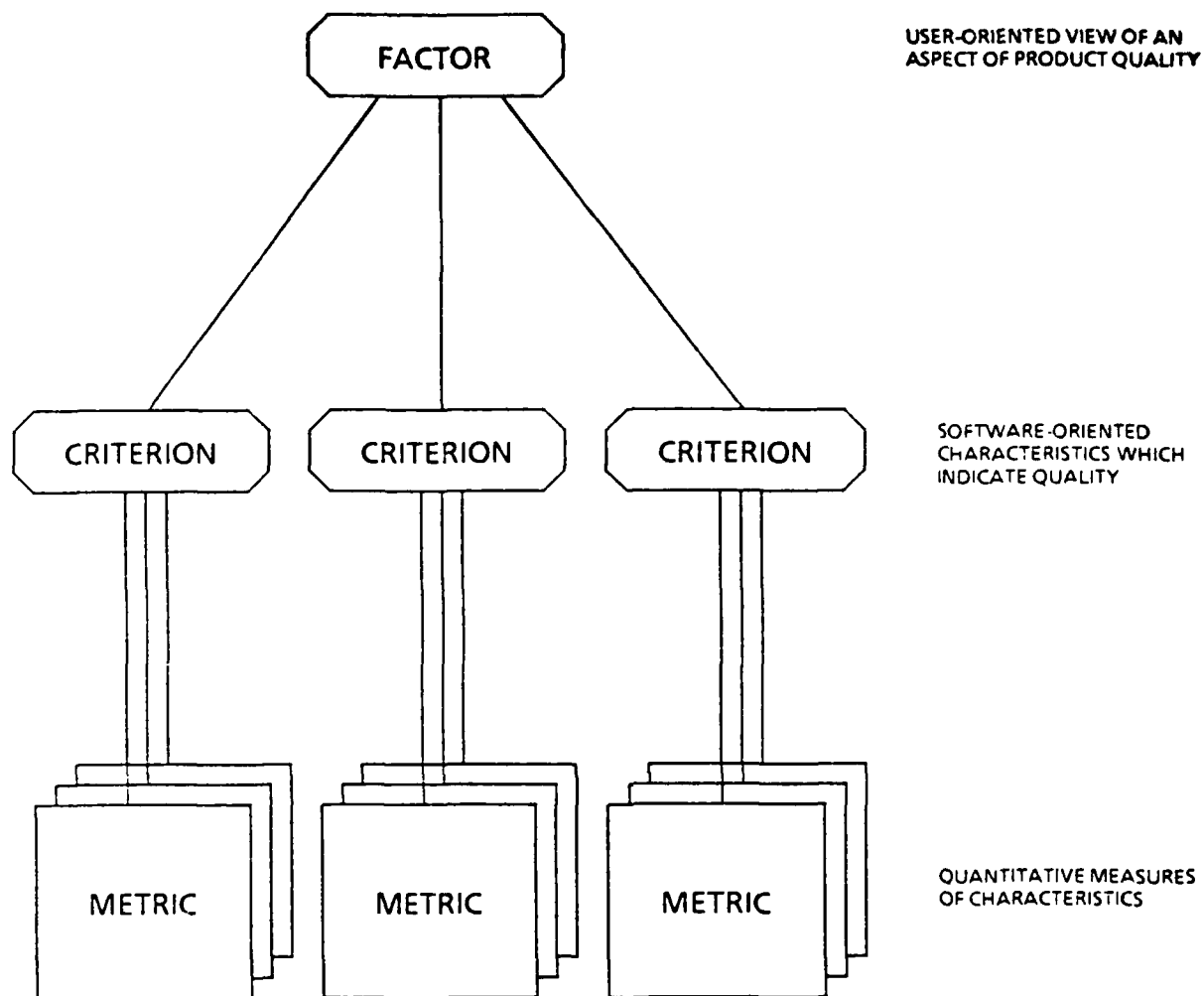


Figure 1.3-1 Software Quality Model

phases and terminology that were used in enhancing the framework and developing the specification methodology. Work on the framework and methodology was accomplished next, using pertinent results from previous RADC contracts concerning software quality measurement. New methodology techniques were developed for performing tradeoffs among quality factors and for relating quality levels to cost over the software life cycle, and metric worksheets (used for collecting metric data) were revised extensively. We prepared a methodology validation plan and made detailed recommendations for changes to Air Force documentation dealing with software quality specification and measurement.

We prepared two guidebooks: Software Quality Specification Guidebook and Software Quality Evaluation Guidebook. Both describe the role of quality metrics within software acquisition management and define all framework elements. The specification guidebook provides a comprehensive set of procedures and techniques for enabling a software acquisition manager to identify and specify software quality factor requirements. The evaluation guidebook provides detailed procedures and techniques to enable data collection personnel to apply quality metrics to software products and to evaluate levels of quality.

1.5 TASK APPROACH AND ACCOMPLISHMENTS

Work was divided into six separate tasks:

- a. Task 1, Develop and Document Approach
- b. Task 2, Enhance Framework
- c. Task 3, Develop Methodology
- d. Task 4, Develop Validation Plan
- e. Task 5, Develop Guidebook
- f. Task 6, Recommend Revisions

Figure 1.5-1 summarizes task interrelationships. Detailed findings for each task are reported in subsequent sections.

Task 1, Develop and Document Approach. The role that quality metrics should play in software acquisition was analyzed. Related concepts were explored: system

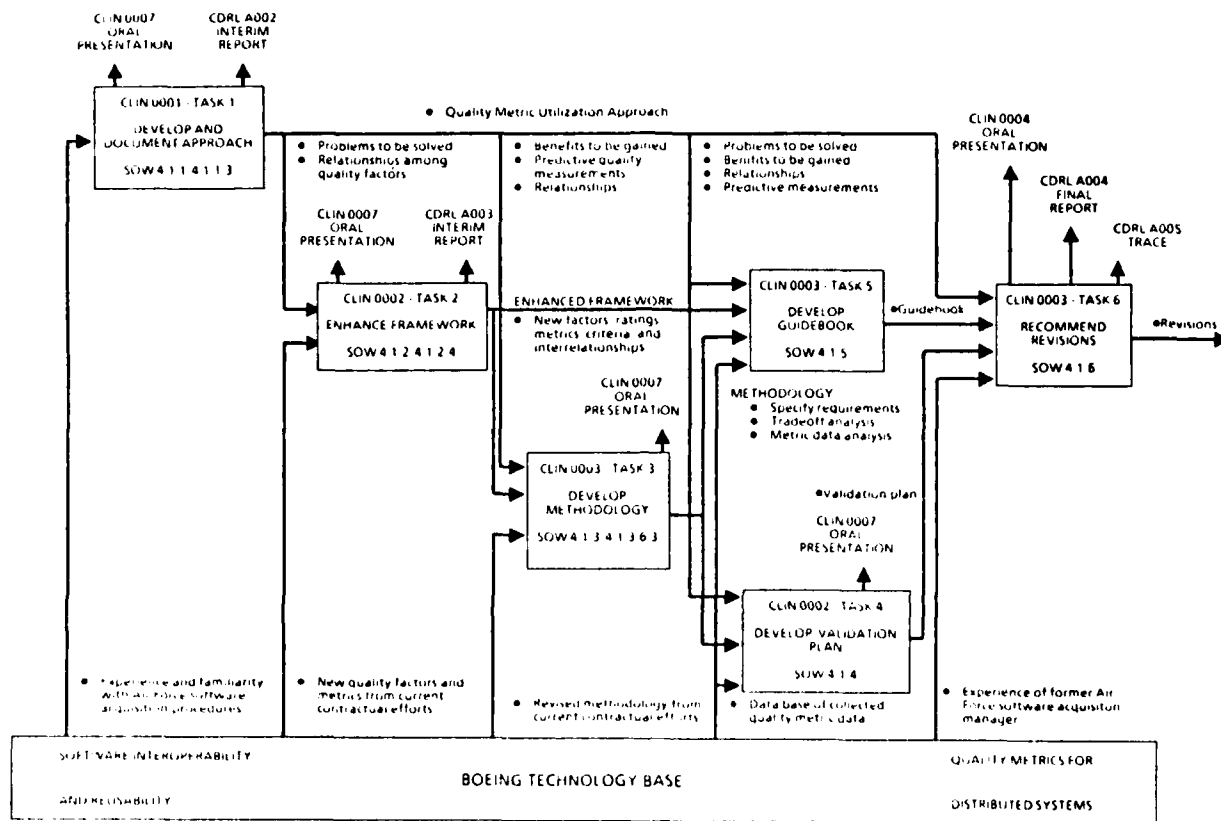


Figure 1.5-1 Task Flow Diagram

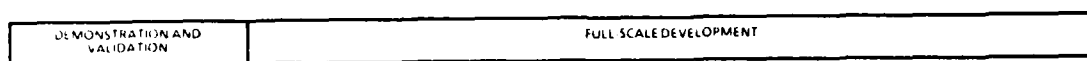
acquisition life cycle, software development cycle, life-cycle relationships, software acquisition management, verification and validation, and QA. DODD 5000.1 and the proposed DOD-STD-SDS (Dec. 1983 draft) were used as baselines for phases and terminology.

A life-cycle model for use with QM technology was defined (shown in Fig. 1.5-2), and recommendations were made for both short-term and long-term technology implementation. Potential benefits and problems with implementation were also explored. Detailed results are reported in Section 2.0. Concepts for the role of quality metrics technology in acquisition management and definitions of life-cycle phases and terminology were used in enhancing the quality metrics framework (Task 2) and developing the quality metrics methodology (Task 3). A candidate list of documentation was used as the initial list for investigating and recommending revisions (Task 6).

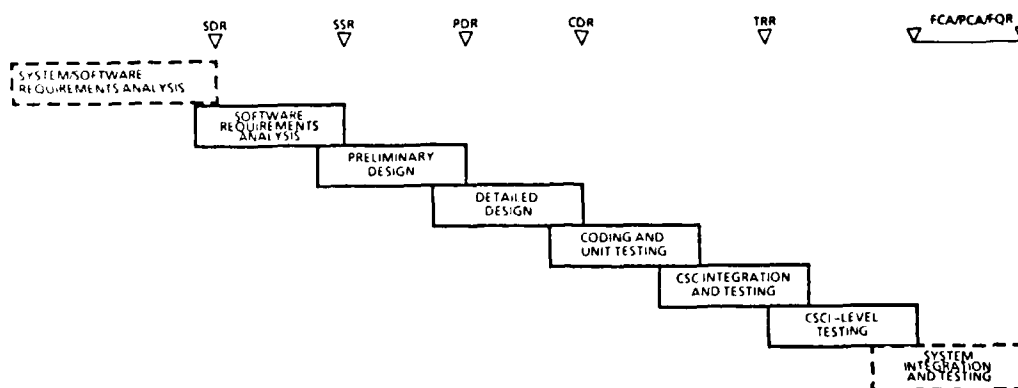
Task 2, Enhance Framework. The framework from the most recent RADC quality measurement contract (described in RADC-TR-83-175) was used as a baseline. Factors were grouped under new acquisition concerns, and new rating formulas were developed for several factors. Minor changes were made to the organization of the criteria and metrics to provide consistency within the framework and to enable more concise specification of requirements. Metric worksheets were revised extensively. One problem with the worksheets was that the metric questions reflected results from four different contracts: styles differed and terminology was inconsistent. Another problem was that worksheets were organized by life-cycle phases different than those defined by the life-cycle model developed in Task 1. Another problem was that information for using the worksheets was documented under three separate covers: the metric questions themselves on worksheets, a set of explanations for understanding metric questions, and a set of tables containing formulas for relating raw data entered on the worksheets.

Goals for revising the worksheets included: consistency and clarity of metric questions and terminology, compatibility with DOD-STD-SDS phases and software terminology, standalone worksheets that required no reference material for answering questions, and consistency with the Software Evaluation Reports proposed as part of the STARS measurement DIDs and being prepared by Dynamics Research Corporation

SYSTEM ACQUISITION PHASES:



Quality Metrics - Software Life Cycle Model



Quality Metrics - Specification



Quality Metrics - Monitoring

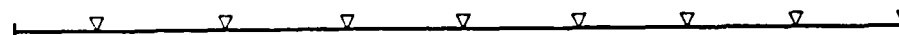


Figure 1.5-2 Quality Metrics Technology - Life-Cycle Model

(DRC). BAC and DRC cooperated in defining and clarifying phases, terminology, and wording of metric questions; explanatory material and examples were included with questions. The BAC format for metric questions included formulas for relating raw data. All goals in revising the worksheets were achieved.

Factor scoresheets were created for evaluating metric data; i.e., translating worksheet data into scores for metric elements, metrics, criteria, and factors. Detailed results are reported in Section 3.0. Factor definitions, interrelationships, and attribute relationships were used throughout procedural steps in the methodology (Task 3).

Task 3, Develop Methodology. The methodology from the most recent RADC quality measurement contract (described in RADC-TR-83-175) was used as a baseline. This baseline was extensively revised and enhanced. The specification process was refocused to enable definition of software quality factor requirements for system and software-unique functions at the system level and to enable allocation of quality requirements to individual computer software configuration items (CSCI) supporting those functions. Trade study techniques and procedural steps were developed to aid in: choosing a set of quality factors, choosing quality-level goals for each factor, revising goals based on beneficial and adverse relationships among factors chosen, revising goals based on projected costs for achieving a quality-level goal over the software life cycle, specifying software quality factor requirements in the system requirements specification, and assessing compliance with those requirements using quality evaluation scores. An example for an airborne radar system is threaded throughout the procedures and techniques.

Techniques and procedures for evaluating the quality level of system and software products were revised and expanded to include using the new metric worksheets for data collection, the new factor scoresheets in data evaluation, and a new software quality evaluation report for reporting data collection and evaluation results. Features are reported in Section 4.0. Procedural steps for the methodology were documented in developing the guidebooks (Task 5).

Task 4, Develop Validation Plan. We prepared a validation plan, which proposes to apply the new methodology to one or more projects, beginning with system and

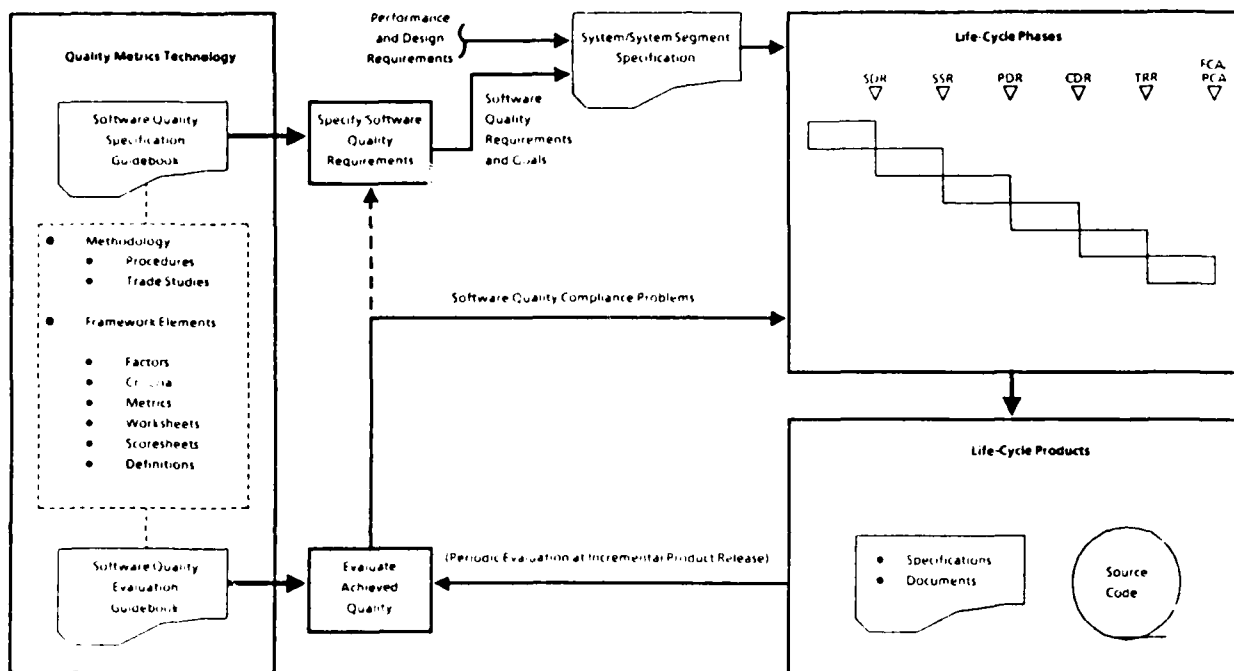


Figure 1.5-3 Software Quality Measurement Methodology

software requirements analysis. A basis for rating the success of the validation effort was outlined. A set of records to be maintained during validation was identified to aid in interpreting validation results. The validation plan is described in Section 5.0.

Task 5, Develop Guidebook. We developed two guidebooks to support the methodology (see Fig. 1.5-3): Software Quality Specification Guidebook and Software Quality Evaluation Guidebook. The specification guidebook provides procedures and techniques to aid the software acquisition manager in identifying the specifying software quality factor requirements and in assessing compliance with those requirements. The evaluation guidebook provides procedures and techniques to enable data collection and analysis personnel to apply quality metrics to software products and to evaluate product quality levels. Both guidebooks describe the role of quality metrics within software acquisition and describe all framework elements. The Software Quality Specification Guidebook is Volume II; The Software Quality Evaluation Guidebook is Volume III.

Task 6, Recommend Revisions. We reviewed documentation used in command and control applications by software acquisition managers. Recommendations were provided for revising selected regulations, standards, DIDs, and guidebooks. Results are reported in Section 6.0.

1.6 CONCLUSIONS

The work performed under this contract is part of an evolving technological trend to provide tools for a more scientific approach to the process of software acquisition management and to the discipline of software engineering. There is an awareness that certain software characteristics influence the quality level and cost associated with a software product. For example, modular and well-commented code is easier to maintain and to reuse. Until now, there has been no organized approach for identifying the many significant variables contributing to low software quality.

The software quality model (Fig. 1.3-1) and software quality factors, criteria, and metrics provide an organized view of different aspects of software quality and of significant software characteristics that contribute to those qualities. Other views of software quality aspects are possible; however, the view presented here, coupled with

the methodology, procedural steps, and mechanisms for collecting and analyzing metric data, provide a wholistic technology for dealing with software quality problems. The software acquisition manager is provided with a powerful tool for communicating quality needs of the user, application, and system; the development contractor is provided a set of quality goals to aid in parameterizing requirements and in making design decisions; both the acquisition manager and the development contractor receive periodic feedback indicating achieved quality levels throughout the development process—providing better management visibility and enabling timely decision making.

2.0 ROLE OF QUALITY METRICS IN THE SOFTWARE ACQUISITION PROCESS

This section summarizes results of Task 1, Develop and Document Approach. Information on task results were originally released as interim technical report 1 (CDRL A002) in December, 1982. Task results were updated with new information during the contract period.

This section examines elements of Air Force system acquisition and software acquisition processes, describes the process used for specifying and monitoring quality levels, and discusses the role of quality metrics (QM) technology in the Air Force software acquisition management process. Considerations include how QM technology can be integrated into the Air Force software acquisition process and how existing mechanisms within the acquisition process can be used to implement QM technology.

Conclusions are that QM technology could be integrated into the Air Force software acquisition process with minimum impact and that existing mechanisms within the acquisition process could be augmented to implement QM technology. Specifically, to specify quality levels and other QM requirements, it is recommended that System Program Office (SPO) software engineering interface with the using command, Air Force Logistics Command (AFLC), and Product Division Software Quality Assurance (QA) organization. And then to monitor the quality levels achieved, it is recommended that one of several groups who already perform a monitoring function (SPO Engineering, Product Division Software QA, Air Force Plant Representative Office (AFPRO), or an independent verification and validation (IV&V) contractor) be tasked with gathering and evaluating metric data. The Data and Analysis Center for Software (DACS) at Rome is recommended as a central repository for metric data.

Recommendations include near-term and long-term activities for implementing QM technology. Near-term activities include changing regulations, standards, and DIDs, selecting trial programs for implementing QM technology, and conducting classes for education and training in QM technology. Recommended long-term activities include selecting and validating metrics appropriate to each product division, establishing an historical data base for QM data, automating portions of QM data collection and analysis tasks, and developing QM analysis tools.

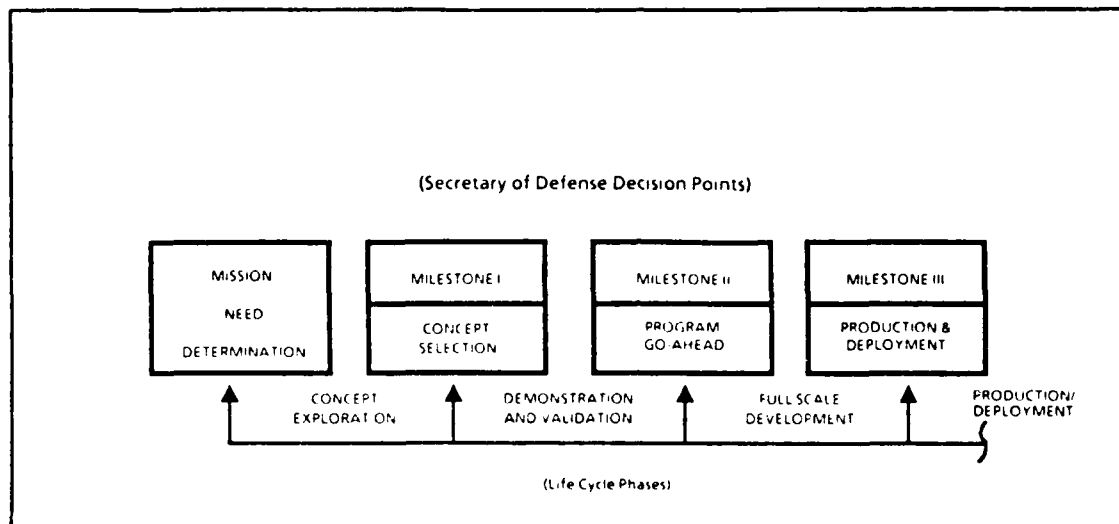


Figure 2.1-1 System Acquisition Life-Cycle Phases and Decision Points

Advantages and disadvantages of using QM technology in software acquisition management and of integrating QM technology into the software acquisition management process are also discussed. Potential benefits include a higher quality end product, better management visibility and control, greater emphasis on quality throughout the life cycle, and life-cycle cost savings. Potential problem areas include maintaining a current QM technology baseline, subjective judgement in scoring some metrics, and lack of manpower for implementing QM technology.

2.1 SOFTWARE ACQUISITION PROCESS

The following sections describe selected concepts associated with Air Force software acquisition management, including system acquisition life cycle, software development cycle, life-cycle relationships, software acquisition management, verification and validation (V&V), and quality assurance (QA). Concepts introduced here provide a basis for discussions of QM technology integration and implementation in the acquisition process in later sections. The system acquisition life cycle and software development cycle are fully defined in DODD 5000.1 and DOD-STD-SDS and are only summarized here. This section is not intended to describe all activities of each life-cycle phase but to establish the background for discussion of the role of QM technology.

2.1.1 System Acquisition Life Cycle

The system acquisition life cycle defined in DOD-STD-SDS consists of four phases: concept exploration, demonstration and validation, full-scale development (FSD), and production and deployment. Four major decision points are associated with these phases as shown in Figure 2.1-1 and as defined in DODD 5000.1 (Major System Acquisition). These points are mission need determination; concept selection, milestone I; program go-ahead, milestone II; and production and deployment, milestone III. The Secretary of Defense, advised by the Defense Acquisition Executive (DAE), decides at these points whether to continue the program and proceed to the next phase or to terminate the program. The system acquisition life cycle applies to the whole system, not the individual parts.

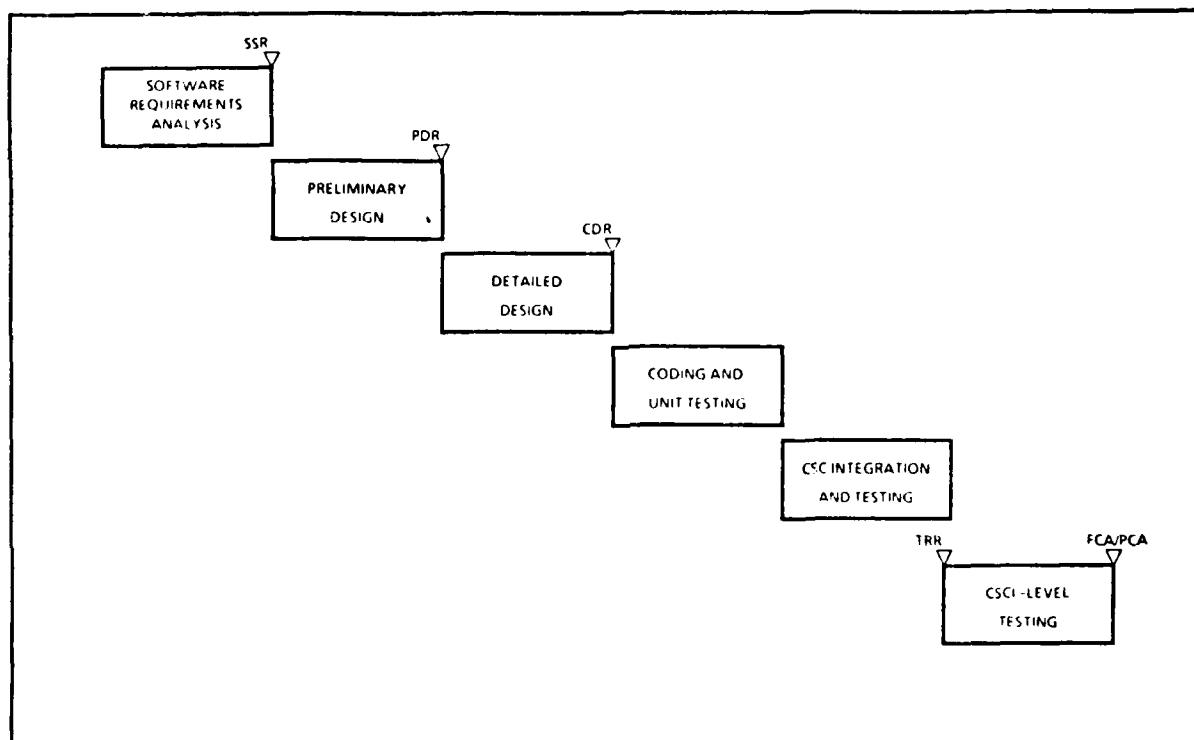


Figure 2.1-2 Software Development Cycle

Concept exploration is the initial planning phase, during which the role of and plans for using computer resources in the system are explored. During demonstration and validation, translating operational requirements into functional, interface, and performance requirements is completed; and requirements for each hardware and software configuration item are defined. During FSD, the system is designed, built, tested, and evaluated. These initial three phases should result in a system meeting specified requirements. Production and deployment includes production (if applicable) and delivery and includes all activities involved in supporting the system until it is retired.

2.1.2 Software Development Cycle

The software development cycle, as defined in DOD-STD-SDS, consists of six phases: software requirements analysis, preliminary design, detailed design, coding and unit testing, computer software component (CSC) integration and testing, and computer software configuration item (CSCI) level testing (see Fig. 2.1-2). This cycle, however, is not standardized and there are many variations throughout the industry. Although names and breakdowns vary, the same process is generally followed.

All software requirements are specified during software requirements analysis. The authenticated software requirements specification (signed off by both the customer and contractor) forms the baseline for preliminary design. During preliminary design, a modular, top-level design is developed from the software requirements. During detailed design, the top-level design is refined to successively lower levels until individual units, which perform single, nondivisible functions, are defined. During coding and unit testing, the designer translates the design approach into code and executes verification tests. During CSC integration and testing, code units are integrated and informal tests are performed on aggregates of integrated units. This cycle concludes with CSCI-level testing, during which formal tests are conducted on the software.

As with the system acquisition life cycle, the software development cycle has decision points associated with most phases. These decision points (shown in Fig. 2.1-2) are the: software specification review (SSR), preliminary design review (PDR), critical design review (CDR), test readiness review (TRR), and functional configuration audit

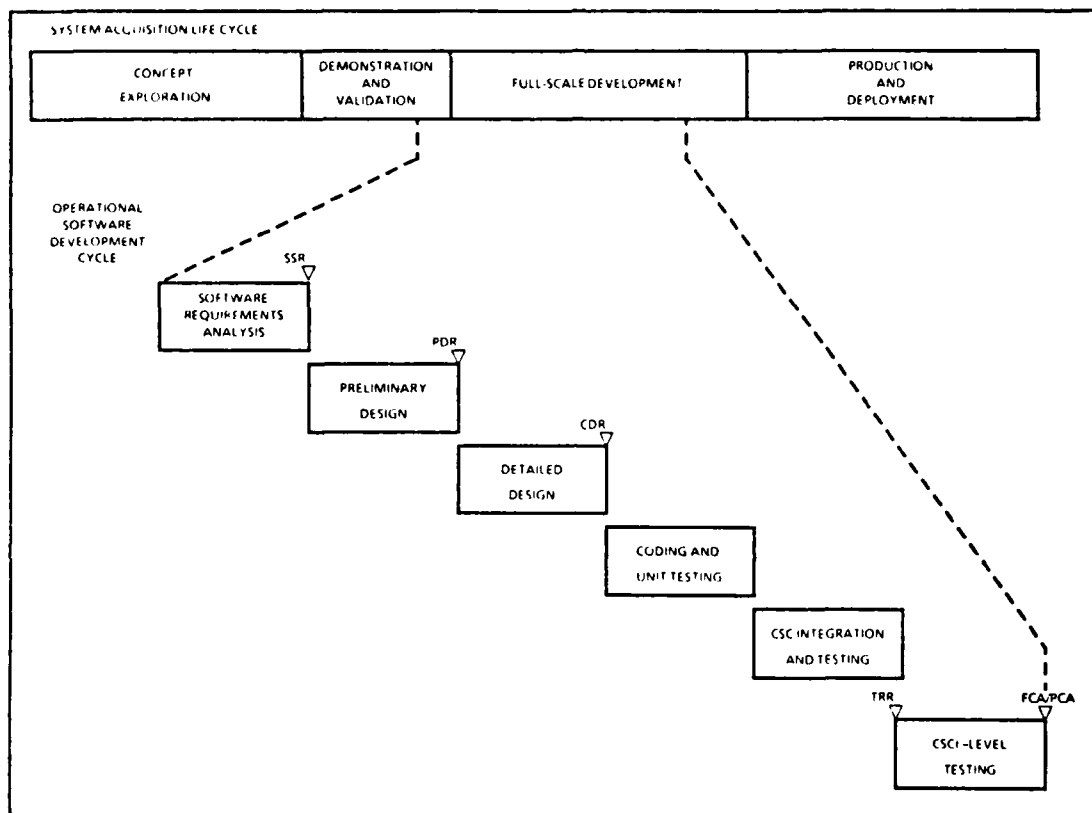


Figure 2.1-3 Life-Cycle Relationship between the System and the Operational Software

(FCA)/physical configuration audit (PCA). These decision points are quite different from decision points associated with the system acquisition life cycle. At these decision points it is not determined whether to continue or terminate the program; rather, progress up to that point is reviewed and it is decided if the developer has completed the current phase and is ready to proceed into the next phase.

2.1.3 Life-Cycle Relationships

Each CSCI to be developed goes through the entire software development cycle. The software development cycle can be completed in a single phase of the system acquisition life cycle or can overlap several phases. For example, software could be developed for risk-reduction analysis during concept exploration or demonstration and validation. This software could be used to validate the feasibility of an algorithm or to compare alternative approaches. This type of software may not be in the language required for the operational software and may not be targeted for the same computer. However, it still goes through the entire development cycle. The same is true for test software developed to aid in validation of the operational software. Operational software development may overlap several system life-cycle phases; requirements definition for operational software begins early in the system acquisition life cycle, although operational software is not fully developed until FSD. In this guidebook operational software quality is the primary concern; therefore, the relationship of the operational software development cycle to the system acquisition life cycle will be examined.

There is a specific relationship between the operational software development cycle and the system acquisition life cycle in most system procurements (see Fig. 2.1-3). The software requirements analysis phase overlaps part of the demonstration and validation phase and the beginning of FSD. The remaining operational software development phases occur during FSD; i.e., preliminary design through CSCI-level testing of the software development cycle. This relationship is assumed for the remaining discussions.

2.1.4 Software Acquisition Management

The software acquisition manager has various responsibilities during the software development cycle. This section focuses on two general functions of software acquisition management: (1) specifying requirements and (2) monitoring development to ensure satisfying the requirements. To describe all that this manager does during the software life cycle is beyond the scope of this guidebook.

Specification of software requirements begins with development of the system specification and continues until all requirements for each CSCI have been specified during software requirements analysis in the software development cycle. These requirements include more than traditional functional and performance requirements. They also include interface, human engineering, language, data base, delivery, self-test, anomaly management, resource reserves, and quality requirements. Many decisions are made to specify these requirements.

The software acquisition manager becomes involved at the system level, when system functional tasks are allocated to software or to hardware. Allocation decisions may be based on trade studies, system engineering, and risk analyses. Once the allocation of functional tasks is completed, specific software requirements can be identified. The result is a set of software capabilities, performance levels, and design constraints. Identification of these specific requirements usually involves decisions supported by trade studies. Such trade studies may include, for example, higher order language (HOL) versus assembly language, distributed processing versus centralized processing, growth capability required for timing and sizing, the degree of human operator interaction required, and efficiency versus maintainability. These software trade studies consider life-cycle costs, risk, schedule, capabilities, software performance, and final product quality. These activities are concluded when the SPO authenticates (signs off) the software requirements specifications for each CSCI.

Once software requirements are specified, the acquisition manager begins monitoring software development. Monitoring continues throughout preliminary design, detailed design, coding and unit testing, CSC integration and testing, and CSCI-level testing and may continue into the system integration and testing that follows. The primary concern of monitoring, other than schedule or cost, is whether the software satisfies

the requirements. Monitoring provides the acquisition manager with visibility of the evolving product in order to track technical progress and quality. This visibility is achieved through various reviews, audits, documentation, and products required periodically throughout development. Established criteria and measurement methods for each review and audit and for all documentation and products are necessary for tracking progress. Tracking enables the manager to identify problems early enough to correct them. Two activities providing feedback are V&V and QA.

2.1.5 Verification and Validation

The purpose of V&V is to provide the Air Force with systematic assurance that acquired software will perform missions in accordance with requirements. The terms verification and validation are often used interchangeably, but in the software development cycle distinct concepts are associated with each. The meaning of these terms as used here is as follows:

Verification is the iterative process of determining whether the product of each software development phase fulfills requirements levied by the previous phase. That is, (1) software requirements are verified to ensure that they fulfill system-level requirements, (2) the software design is verified to ensure that it satisfies requirements in the software requirements specification, and (3) code is verified to ensure that it complies with the top-level design and detailed design documents. This process does not consider whether system-level software requirements are correct or whether they actually satisfy users needs.

Validation is a continuing process to ensure that requirements at various levels are correct, thus satisfying mission requirements defined by the using command. Sometimes validation is considered to be the system-level test activity that validates the CSCI against software and system requirements. In reality, it is much more than that. Validation, like verification, continues throughout the software life cycle. For example, when software requirements are allocated and derived, a system-level requirement could be found to be vague or incorrect; or during design, it could be discovered that a software requirement is infeasible or ambiguous. Feedback to the manager enables corrective action to be taken early in development, thereby reducing risk and cost.

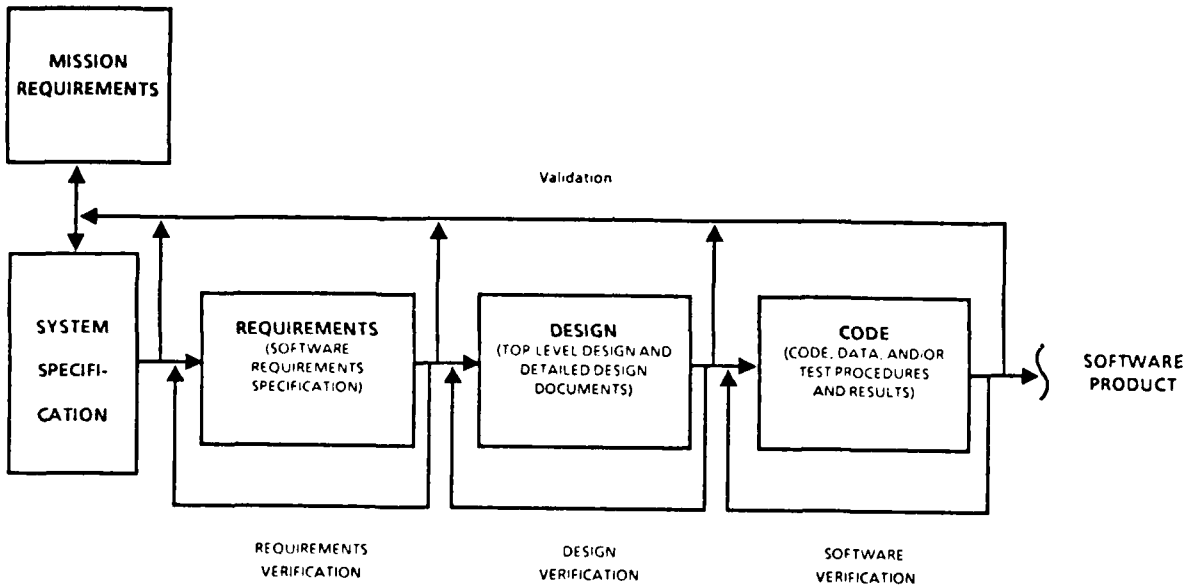


Figure 2.1-4 Relationship of Software Development and V&V

The concept of V&V and its relationship to software development products is shown in Figure 2.1-4. V&V provides feedback to the software acquisition manager concerning software technical performance. The term IV&V is used when V&V is done for the Air Force by a contractor other than either the prime contractor or the subcontractor who is developing the software.

2.1.6 Quality Assurance

According to MIL-S-52779A, the purpose of software QA is to ensure that the software delivered under a contract complies with contract requirements. This type of QA program will not ensure development of a high-quality software product unless software quality attributes are specified in measurable terms as part of the contract. The objective of current QA programs is to provide feedback to the acquisition manager concerning various aspects of the development process. QA is similar to V&V, the major difference being that V&V provides technical feedback on software products at only a few points in time, whereas QA provides feedback on a wide range of development activities. But contractual software quality is not normally defined in quantitative terms. The current goal is simply to achieve better quality through controlling the development processes.

Section 2.3 explores how QM technology can help to expand the scope of QA programs to include specification of software quality requirements and measurement of achieved quality levels for software development products. The following paragraphs explain the current scope of QA programs.

At one time, software QA was equated to testing. As an illustration, Section 4 of the CPCI development specification (according to MIL-STD-483) was called Quality Assurance Provisions. However, as with other products, it was learned that quality cannot be tested into software. Because of cost and schedule impacts, it is usually too late to make changes when quality problems are found during testing. Quality can be affected by how code is written and how software is designed. If a software quality problem is found during testing, it is usually very expensive to redesign and to change the code. Quality should be planned, designed, and built into software. This realization has led to the current life-cycle-oriented QA approach. This approach focuses attention on all phases of the software development cycle; and software QA

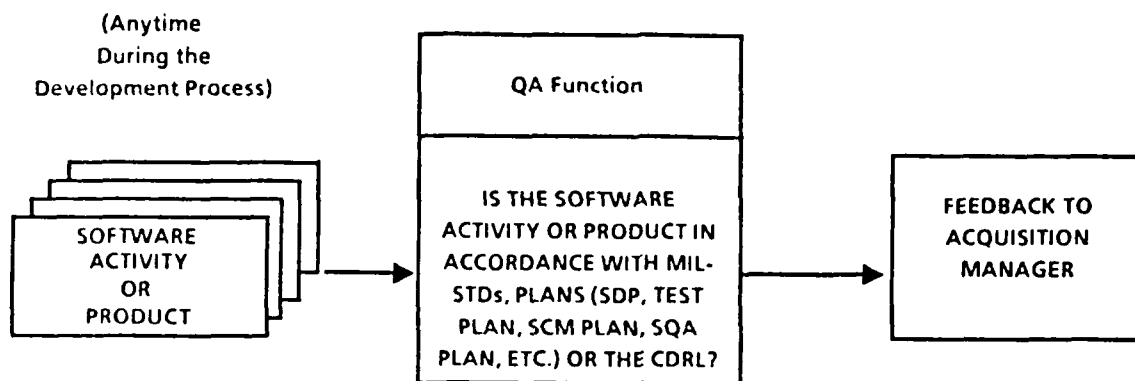


Figure 2.1-5 Software QA Function

now includes many activities, such as ensuring that software is being developed in accordance with plans, that requirements are traceable, that design and code are easily and economically supportable, and that testing is accomplished as planned. These activities provide necessary feedback to the software acquisition manager.

Software quality assurance programs, however, are primarily administrative rather than technical. For example, the QA organization does not trace requirements but ensures that Engineering has developed traceability matrices. The QA function is essentially a checkoff function applied during the software development process; i.e., QA ensures that everything is done as planned. Software QA continues throughout the software development cycle (see Fig. 2.1-5).

Software QA is an evolving discipline. Experience has provided insight into which development practices tend to produce a higher quality software product, and the QA program ensures that selected practices are used by checking the development process. The next step to improving quality is to quantitatively specify quality requirements and to measure and control the quality of the software product as it evolves. Implementing QM technology in the Air Force acquisition process will provide the added dimension of quantitative measures to addressing quality concerns for software products.

2.2 QUALITY METRICS

The purpose of QM technology is to enable the software acquisition manager to specify a desired software quality level for each quality factor of importance to the application and to quantitatively measure the achieved levels of quality at specific points during development. These periodic measurements enable an assessment of current status and a prediction of quality level for the final product. Some problems with delivered software products have been that these products are (to varying degrees) unreliable, incorrect, and/or unmaintainable. QM technology addresses these and other quality-oriented problems by providing a means to specify quality requirements, to quantitatively measure quality achieved during development, and to predict a quality level for the final product.

QM technology measures the degree of software quality, not the level of software technical performance; e.g., how easy is it to maintain the software, not how accurate is the navigation algorithm. However, the process of specifying and measuring quality levels is analogous to the process of specifying and measuring technical performance. Both processes begin with similar activities: system needs are assessed, trades are performed (involving resources and levels of performance or levels of quality), and requirements are specified. Subsequent phases involve evaluations of how well these requirements are being satisfied.

Technical performance levels are traditionally evaluated by modeling in early development stages and by testing in later development stages. Quality has traditionally been evaluated by such methods as reviews, walk throughs, and audits. This type of quality evaluation ensures that, for example, designs are traceable to requirements, configuration management is adequate, and standards and plans are being followed. However, it does not address such quality issues as software reliability, correctness, and maintainability. QM technology enables a quantitative assessment of these types of quality factors at different stages of development, thereby ensuring that specified quality levels are being satisfied in a manner similar to performance evaluation by testing.

Figure 1.5-2 depicts the software life-cycle model used in QM technology. The software model is shown in typical relationship to two system acquisition phases. Eight development states are shown with typical review and audit points. There are two system-level activities involving software: system/software requirements analysis and system integration and testing (both shown in dashed boxes). (Operational testing and evaluation is the last FSD phase but is not shown as it is not normally performed by the development contractor.) There are six software development phases: software requirements analysis, preliminary design, detailed design, coding and unit testing, CSC integration and testing, and CSCI-level testing. These phases refer to the same development activities as are described in Section 2.1. This division of activities was chosen because at the end of each activity shown in Figure 1.5-2 a configuration baseline generally is established, and software products (specifications, documents, code) describing that baseline are available for review or audit and the application of quality measurements. Also illustrated in Figure 1.5-2 are the two points at which quality requirements are specified and the eight points at which

quality levels are measured (monitored). These measurement points generally correspond to the review or audit points for configuration baselines.

2.2.1 Framework

A hierarchical model for quality has been established (see Fig. 1.3-1). User-oriented factors (e.g., reliability, correctness, maintainability) are at the top level, software-oriented criteria are at the next level, and metrics—quantitative measures of characteristics—are at the lowest level.

This model is flexible in that it indicates a general relationship between each factor and its attributes. This permits updating of individual elements to reflect technology advances without affecting the model itself. For example, as new user concerns evolve, new factors can be added at the top level; and as software technology evolves, criteria and metrics can be added, deleted, or modified as necessary. There are currently 13 quality factors, 29 criteria, 73 metrics, and more than 300 metric elements (distinct parts of a metric). Table 2.2-1 shows the 13 quality factors and describes the primary user concern for choosing each factor. Quality factors and user concerns are categorized by three types of acquisition concerns with respect to the software: (1) product performance—how well does the software function in its normal environment; (2) product design—how valid (appropriate) is the design with respect to requirements, verification, and maintenance; and (3) product adaptation—how easy is it to adapt the software for use beyond its original intended use (e.g., for new requirements, a new application, or a different environment).

Figures 2.2-1, 2.2-2, and 2.2-3 show the quality factors, criteria, and metrics in the hierarchical relationships of the software quality model. The metrics are identified by acronym only in the figures. These and other framework elements for QM technology are described in detail in Section 3.0. The following sections describe some aspects involved in specifying and monitoring software quality using QM technology.

2.2.2 Quality Specification

When determining and specifying software quality requirements, system needs are assessed from a quality perspective; the desired quality factors, associated criteria,

Table 2.2-1 Quality Concerns

Acquisition Concern	User Concern	Quality Factor
PERFORMANCE - HOW WELL DOES IT FUNCTION?	HOW WELL DOES IT UTILIZE A RESOURCE? HOW SECURE IS IT? WHAT CONFIDENCE CAN BE PLACED IN WHAT IT DOES? HOW WELL WILL IT PERFORM UNDER ADVERSE CONDITIONS? HOW EASY IS IT TO USE?	EFFICIENCY INTEGRITY RELIABILITY SURVIVABILITY USABILITY
DESIGN - HOW VALID IS THE DESIGN?	HOW WELL DOES IT CONFORM TO THE REQUIREMENTS? HOW EASY IS IT TO REPAIR? HOW EASY IS IT TO VERIFY ITS PERFORMANCE?	CORRECTNESS MAINTAINABILITY VERIFIABILITY
ADAPTATION - HOW ADAPTABLE IS IT?	HOW EASY IS IT TO EXPAND OR UPGRADE ITS CAPABILITY OR PERFORMANCE? HOW EASY IS IT TO CHANGE? HOW EASY IS IT TO INTERFACE WITH ANOTHER SYSTEM? HOW EASY IS IT TO TRANSPORT? HOW EASY IS IT TO CONVERT FOR USE IN ANOTHER APPLICATION?	EXPANDABILITY FLEXIBILITY INTEROPERABILITY PORTABILITY REUSABILITY

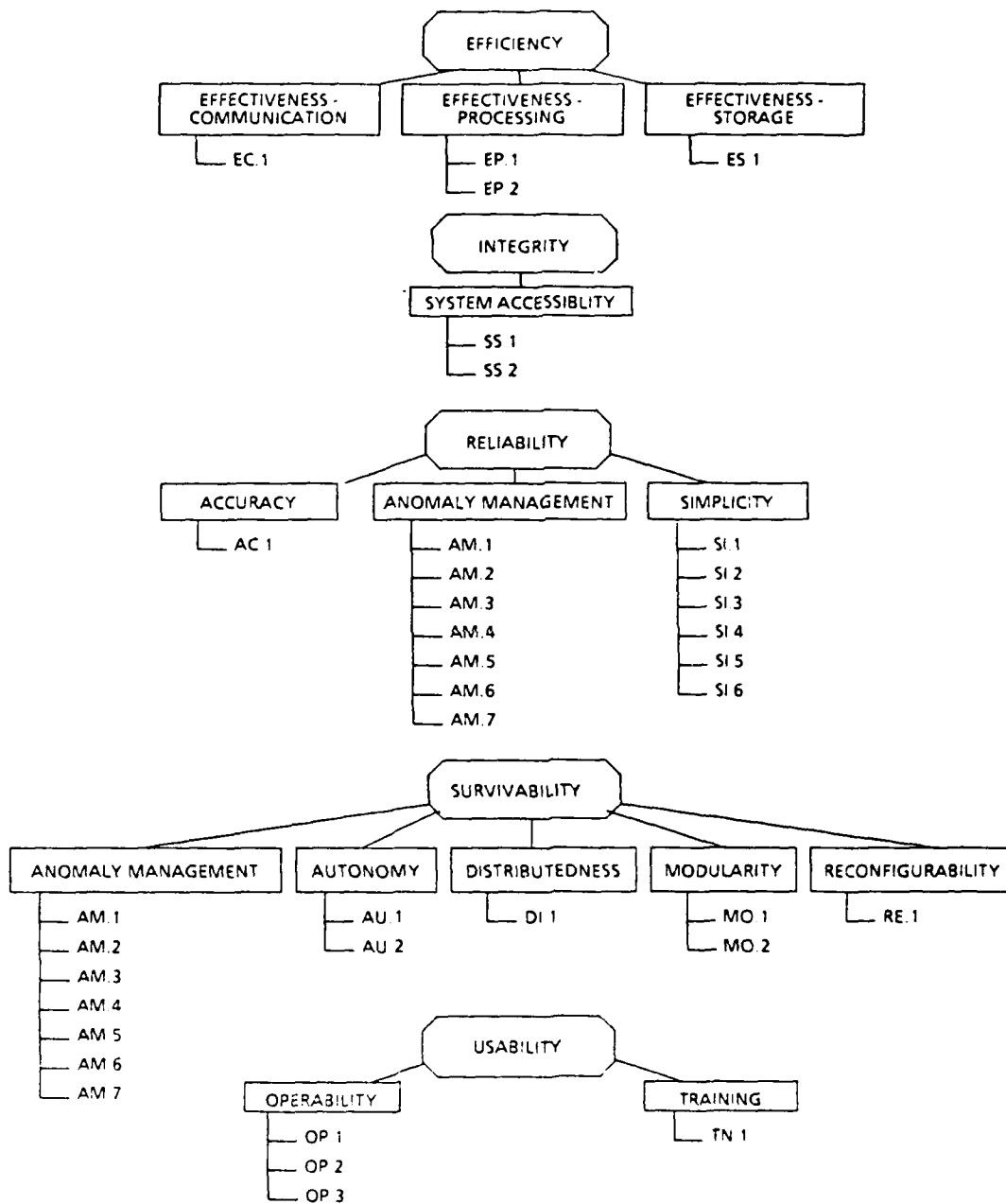


Figure 2.2-1 Performance Factor Attributes

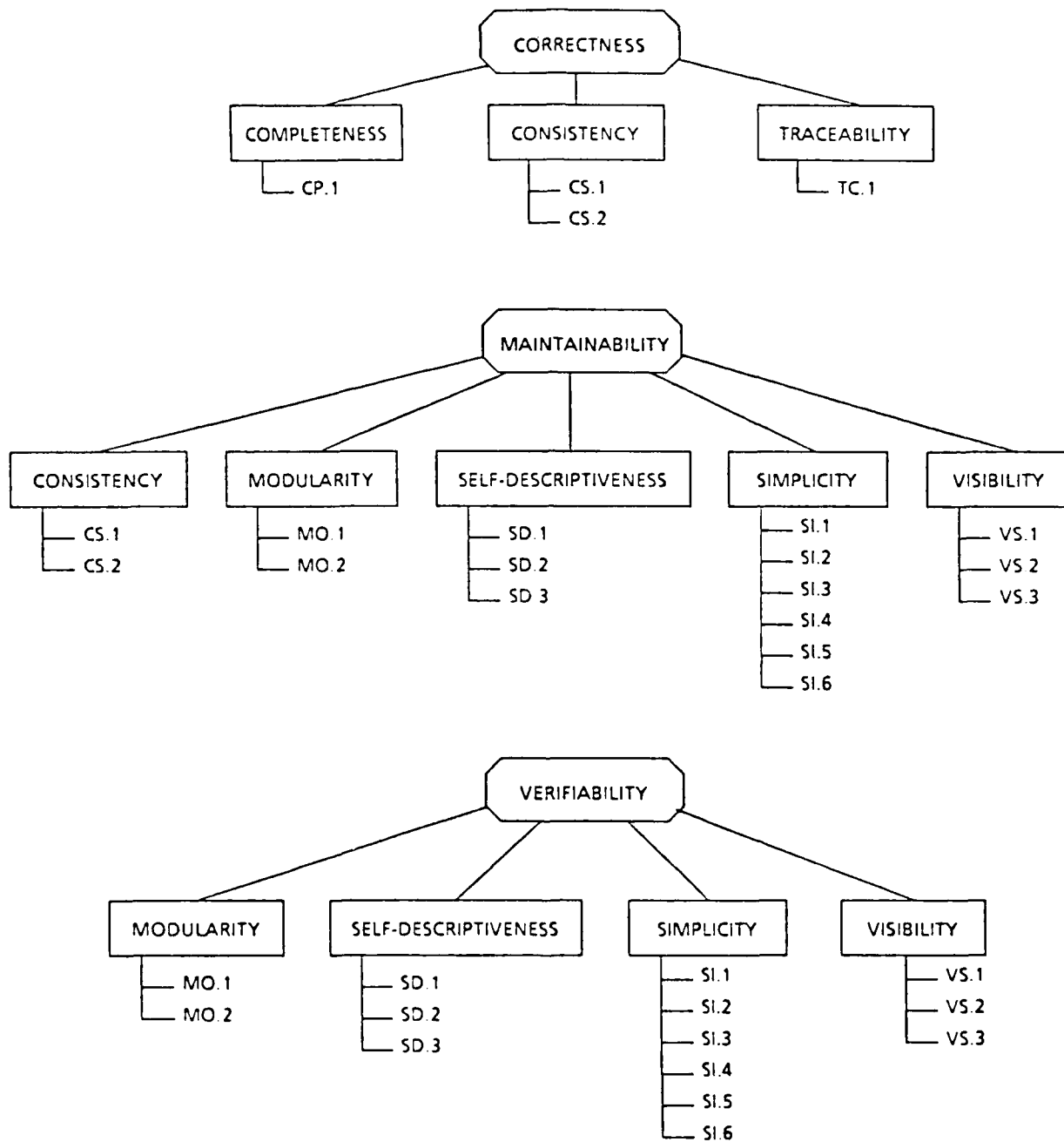


Figure 2.2-2 Design Factor Attributes

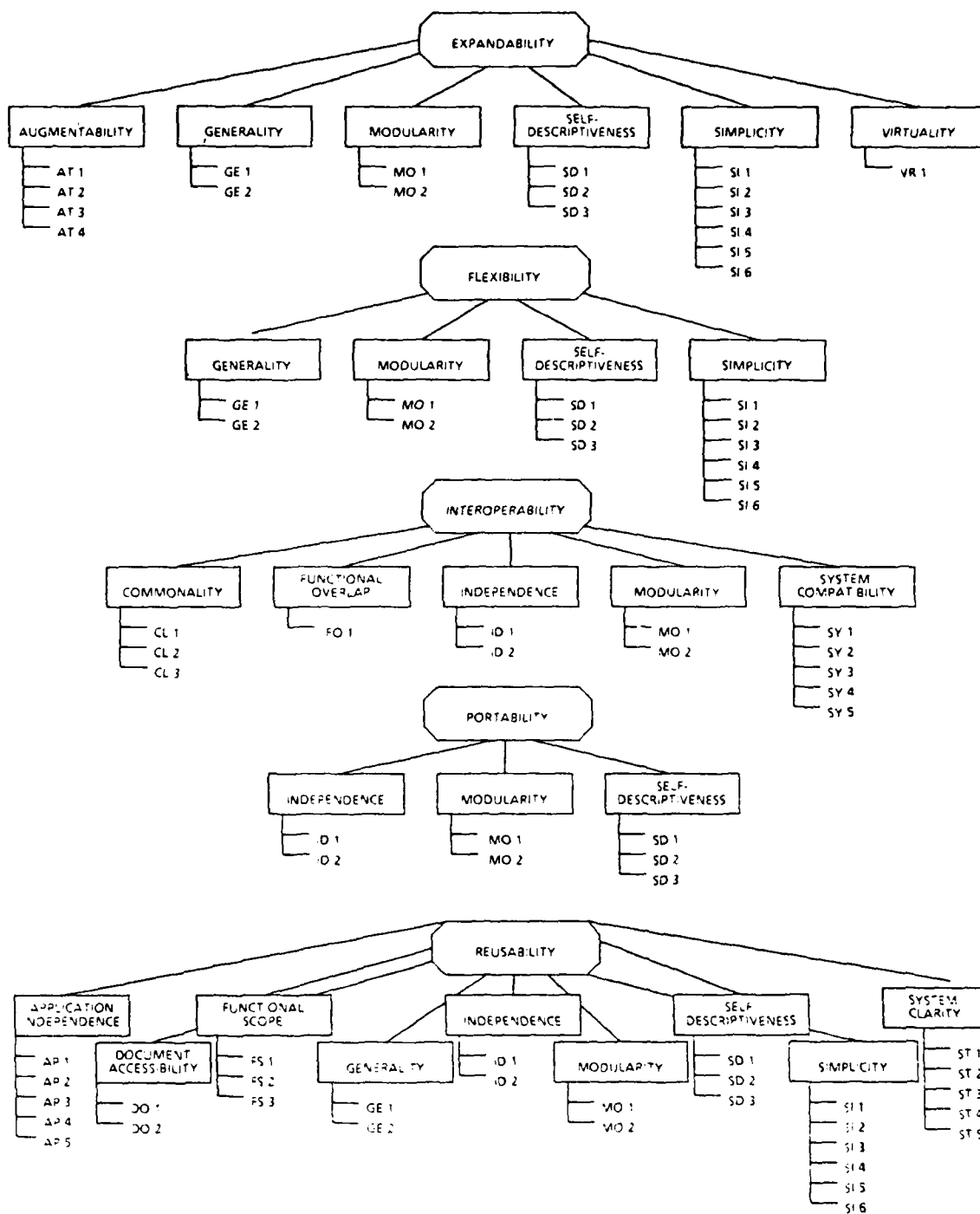


Figure 2.2-3 Adaptation Factor Attributes

Table 2.2-2 Software Quality Factor Interrelationships

ACQUISITION CONCERN		PERFORMANCE					DESIGN			ADAPTATION				
ACQUISITION CONCERN	QUALITY FACTOR AFFECTED	EFFICIENCY	INTEGRITY	RELIABILITY	SURVIVABILITY	USABILITY	CORRECTNESS	MAINTAINABILITY	VERIFIABILITY	EXPANDABILITY	FLEXIBILITY	INTEROPERABILITY	PORTABILITY	REUSABILITY
	QUALITY FACTOR SPECIFIED													
PERFORMANCE	EFFICIENCY	▨						▽	▽				▽	
	INTEGRITY	▽	▨											
	RELIABILITY	▽		▨		△								
	SURVIVABILITY	▽	▽		▨	△		△		▽	▽		▽	▽
	USABILITY	▽				▨		△	△					
DESIGN	CORRECTNESS						▨	△	△	△	△			△
	MAINTAINABILITY	▽						▨	△	△	△			△
	VERIFIABILITY	▽							▨					
ADAPTATION	EXPANDABILITY	▽	▽	▽	▽					▨		△		
	FLEXIBILITY	▽	▽	▽	▽						▨	△		
	INTEROPERABILITY	▽	▽									▨		
	PORTABILITY	▽											▨	
	REUSABILITY	▽	▽	▽	▽			△				△	△	▨

△ = POSITIVE EFFECT

▽ = NEGATIVE EFFECT

BLANK = NONE OR APPLICATION DEPENDENT

and applicable metrics are selected; and quality-level goals are derived for each separate quality factor. When assessing system needs, application characteristics should be considered. For example, if the system will have a long life cycle, emphases on maintainability, flexibility, portability, and expandability are recommended. Factor goals define the required quality levels to be achieved for the factor (i.e., excellent, good, or average). In general, choosing a higher quality goal will result in more resources being expended to achieve that level. When deriving factor goals, interrelationships between factors should be considered because a high quality goal for one factor may conflict with a high quality goal for another factor. Table 2.2-2 shows the beneficial and adverse relationships between quality factors; some factors have a positive relationship and others conflict. For example, specifying a high quality level for most factors conflicts with specifying a high quality level for efficiency.

A typical problem for an embedded software system arises when reliability is of the utmost importance because of the type of mission to be performed, but efficiency is also required because of space and weight limitations, and flexibility is needed because of the variety of missions and/or targets. It is normally infeasible to select and achieve high quality levels for all three factors. Highly efficient code is usually tightly written assembly-level code and tends to be not as reliable or as amenable to changes (flexible) as looser, more structured higher order language code. And code written to be reliable and flexible tends to be less efficient. Trade studies are needed to resolve these problems. If some efficiency is sacrificed for reliability, then performance goals (e.g., for accuracy or range) may be affected. If some flexibility is sacrificed for efficiency, then the scope of the missions and/or targets may be reduced. QM technology provides an aid for decision making when selecting quality-level goals, when determining feasible software requirements, and for allocating acquisition resources. Several iterations of quality tradeoffs may be required for choosing reasonable quality goals. Section 4.0 of the specification guidebook (Vol. II) provides specific techniques for choosing quality factors and includes consideration of application characteristics and factor interrelationships.

2.2.3 Quality Monitoring

When monitoring software quality, the quality metrics (in the form of questions on worksheets) are applied to software products (specifications, documents, code) at different stages of the development cycle, and a quality-level score is calculated for each factor. The factor score predicts a quality level for the final product. The points in the development cycle where data gathering and analysis are recommended is shown in Figure 1.5-2. These points generally correspond to normal reviews and audits conducted when a configuration baseline has been established (SDR, SSR, PDR, CDR, TRR, and FCA/PCA). Before each review or audit, the metrics selected for the project are applied to software products resulting from that phase of development. This results in a quantitative value for each metric. The metric values are then used to calculate scores for each criterion, and the criteria scores are used to calculate a score (predicted quality level) for each factor.

The quality metrics are applied at incremental points during the development phases. This enables periodic review of progress in meeting quality goal requirements and aids in pinpointing areas of weakness (and strength) in product quality as the product evolves. There are two types of metrics—*anomaly detecting* and *predictive*. Both are used in scoring. A low score for predictive metrics indicates that a low score will probably result for the end product because the design is not considering aspects important to achieving the desired quality level. For example, if the design has very little spare storage capacity, the end product will not be highly expandable. A low score for anomaly-detecting metrics indicates an actual design or code deficiency. For example, if provisions are not made for immediate indication of an access violation, software integrity would be jeopardized. Evaluating low metric scores provides an opportunity for identifying deficiencies and anomalies during development when they are more easily corrected.

Worksheets have been devised to help gather metric data. There is a separate worksheet for each development phase, and each worksheet lists only metrics applicable to that phase. A more detailed explanation of the worksheets is provided in Section 3.4.

2.3 SOFTWARE ACQUISITION USING QUALITY METRICS

Two general functions of the software acquisition manager are described in Section 2.1.4: (1) specifying requirements and (2) monitoring development to ensure that requirements are being satisfied. Also two general functions associated with QM technology are described in Sections 2.2.2 and 2.2.3: (1) specifying quality requirements and (2) monitoring development to ensure that metric scores are predicting specified quality goals. When using QM technology, monitoring begins earlier in the development cycle. The relationship of these functions to the software life cycle is shown in Figure 2.3-1.

Specifying and monitoring have not usually overlapped. The specification of software requirements was normally completed before development monitoring began, as shown in Figure 2.3-1. Metric questions have been devised to enable evaluation of software quality reflected in the system specification available at the system design review (SDR). This moves the start of monitoring forward so that the two functions overlap.

Several organizations normally are involved in performing these two functions. Although the internal structure of the Air Force product divisions (ESD, ASD, and SD) may differ, the relationship of the SPO to external organizations is basically the same for each division. Organizations that may be involved in the QM functions and their recommended relationships are shown in Figure 2.3-2. Organizational relationships are discussed in the following paragraphs.

Several organizations should be involved in the specification function. The primary organization responsible for software requirements specification is SPO Software Engineering. However, SPO software engineers need help from both the using command and Air Force Logistics Command (AFLC) to fully define software quality needs. Both organizations have a vested interest in requirements affecting system operation and support.

The using command is primarily interested in operational requirements and is especially qualified to contribute to a definition of quality needs for the performance quality factors (e.g., efficiency, integrity, and reliability). AFLC is primarily interested in support requirements and is especially qualified to contribute to a

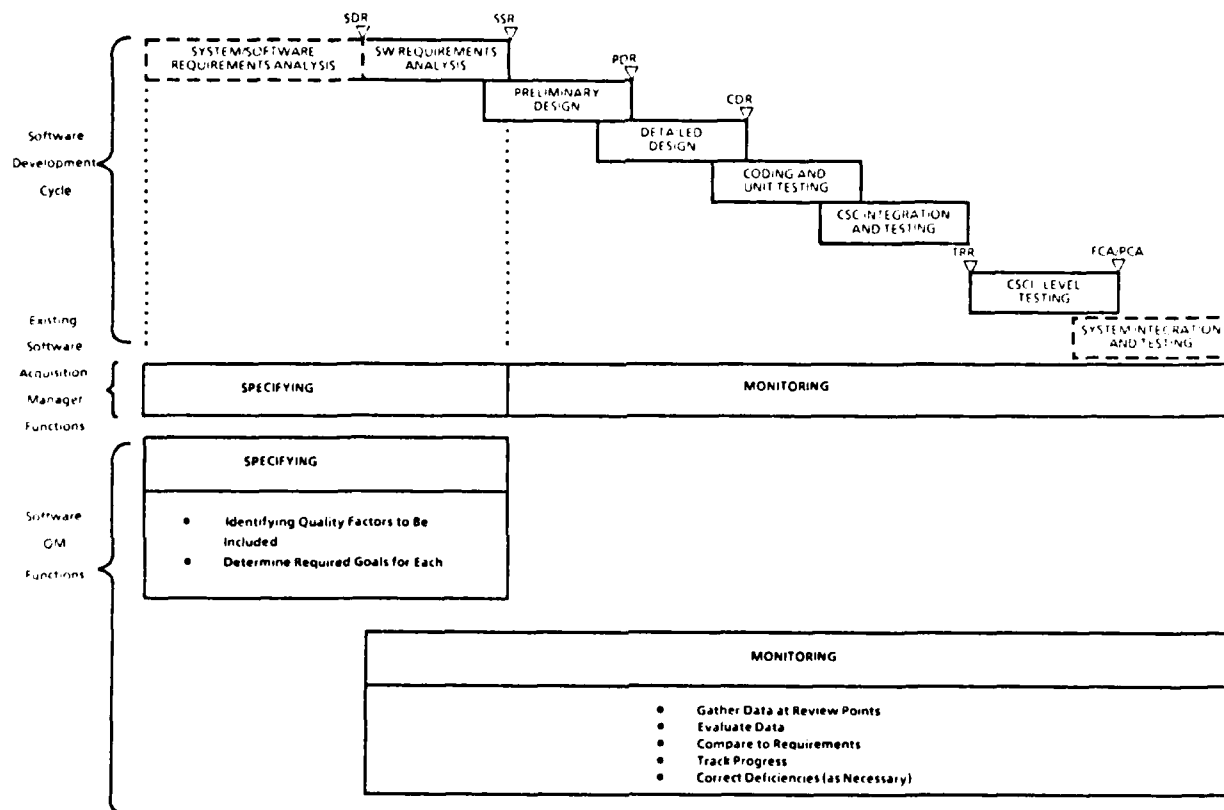


Figure 2.3-1 Software Acquisition Quality Metrics Functions

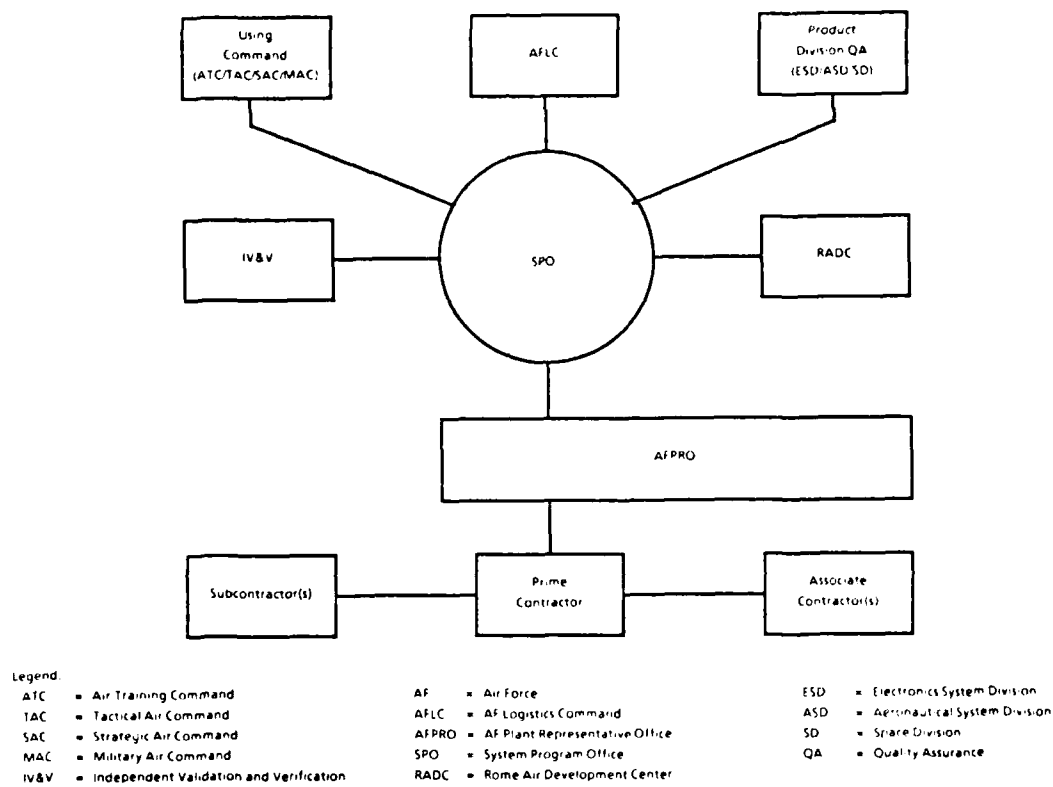


Figure 2.3-2 Air Force Acquisition Relationships Involved in Quality Metrics Functions

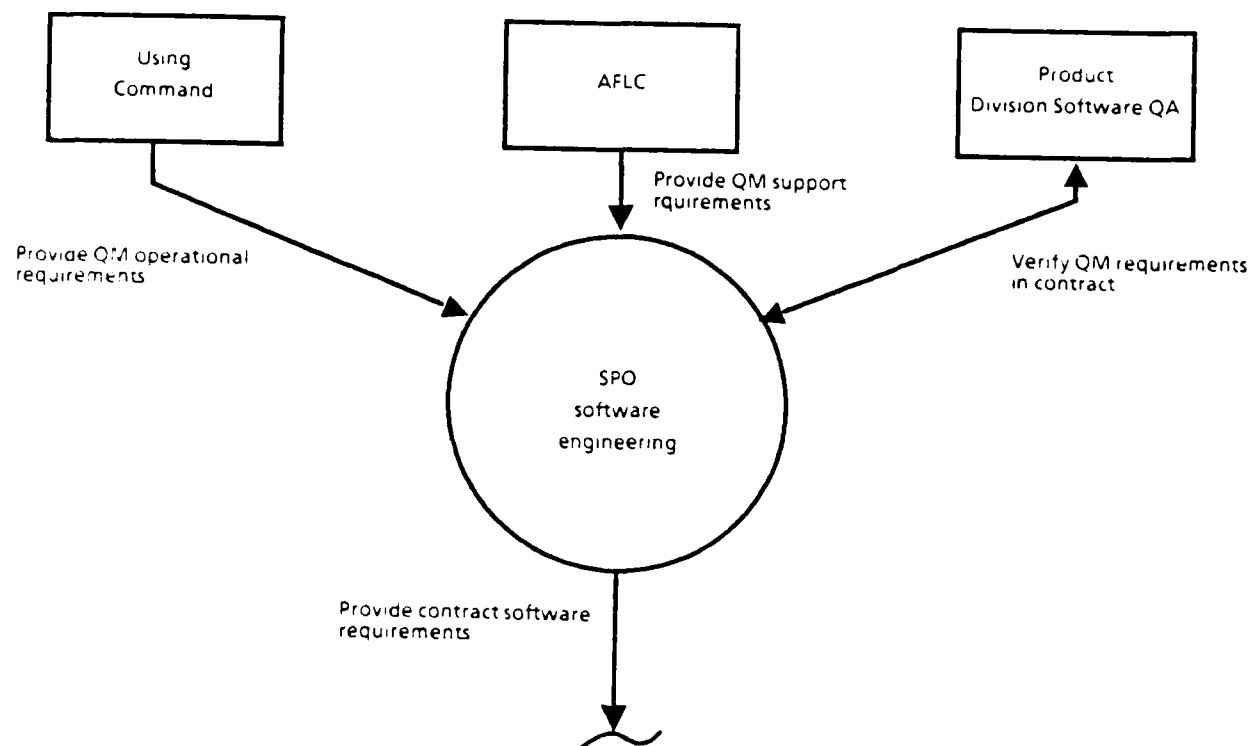


Figure 2.3-3

Recommended Responsibilities and Relationships for the QM Specification Function

definition of quality needs for the design and adaptation quality factors (e.g., maintainability, expandability, and portability). With input from these organizations, SPO Software Engineering can determine the contractual statement of quality requirements. In addition, the Product Division Software QA organization is normally tasked to ensure that quality requirements are included in the contract. These responsibilities and relationships for the specification function are shown in Figure 2.3-3.

Several organizations also should be involved in the monitoring function. Among the first activities are identifying and negotiating with the organization that will collect and analyze metric data. If that organization is to be another Air Force agency, such as Air Force Contracts Management Division (AFCMD), then the SPO needs to negotiate the effort through a memorandum of agreement. If the organization is to be an IV&V contractor, then the IV&V contract needs to be negotiated. These negotiations must be completed very early in the program before data collection starts, and SPO Software Engineering must ensure that necessary support is provided.

Several organizations could collect and analyze data, including SPO Software Engineering, the Product Division Software QA, the Air Force Plant Representative Office (AFPRO), and an IV&V contractor. The following criteria were established to aid in selecting an organization: technical capability, labor availability, economy, and data availability. Technical capability refers to the depth of technical understanding of software by people in the organization. Labor availability refers to availability of qualified people to perform this additional task (i.e., currently available or readily obtainable). Economy refers to the least costly method for the SPO to obtain data. Data availability refers to the ability to access the most current contractor documentation and information. Informal lines of communication greatly influence this factor.

We rated four candidate organizations using these criteria, based on our experience. A score of 1 represents the best conditions and a 3 represents the worst for each criterion. A total unweighted score was determined for each organization, with the lowest score representing the best choice. The evaluation scores are shown in Table 2.3-1.

Table 2.3-1 Organizational Evaluation

<div>CRITERION</div> <div>ORGANIZATION</div>	TECHNICAL CAPABILITY	LABOR AVAILABILITY	ECONOMY	DATA AVAILABILITY	SCORE* SUMMARY
SPO ENGINEERING	2	2	1	2	7
PRODUCT DIVISION SOFTWARE QA	3	3	1	3	10
AFPRO	2	2	1	1	6
IV&V	1	1	3	2	7

1 = BEST
2 = MEDIUM
3 = WORST

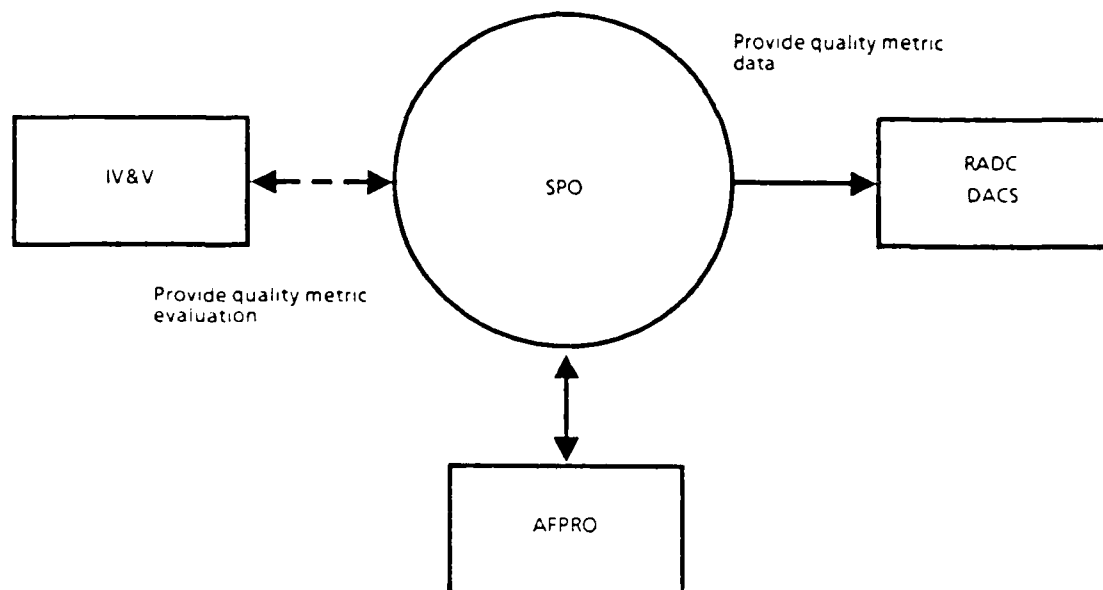
* Lowest Score is Best (Unweighted)

Several assumptions were made for scoring. The first was that all criteria are weighted equally; actually, however, technical capability and labor availability may be overriding factors for selection. For technical capability, it was assumed that Product Division Software QA groups are unlikely to be able to obtain people experienced in both software engineering and QA to perform that job. For economy, it was assumed that any Air Force person (civilian or military) is a free resource for the SPO. Otherwise, the SPO must pay for IV&V contractor services. Data availability scores include the assumption that the IV&V contractor works for SPO Software Engineering and that good communication channels are established. These assumptions may not be valid in all situations.

The AFPRO received the lowest score and, therefore, was rated best. It is generally recommended that the AFPRO perform data collection and analysis for the SPO. When this cannot be negotiated, it is recommended that an IV&V contractor be assigned this task. Although SPO Software Engineering and the IV&V contractor are rated equally, the recommendation to use an IV&V contractor was made because of better labor availability. It is recommended that a chart similar to the one shown in Table 2.3-1 be developed early in a program.

A proposed DID, Software Quality Evaluation Report, is contained in Appendix C and can be used to report data collection and analysis results to the software acquisition manager. This feedback enables the manager to track progress, ensure that requirements are being satisfied, and take corrective action when necessary. Recommendations for responsible organizations and relationships for monitoring are shown in Figure 2.3-4.

The preceding paragraphs discuss government monitoring only, and the development contractor was not mentioned. Because quality factor requirements are included as contractual requirements, the development contractors must also monitor achieved quality levels to show compliance. However, to ensure that data and reports received by the SPO are unbiased, we recommend that the government independently monitor achieved quality levels.



--- indicates alternate source

Figure 2.3-4

Recommended Responsibilities and Relationships for the QM Monitoring Function

2.4 IMPLEMENTING QUALITY METRICS

We recommend performing both near-term and long-term activities to ensure successful implementation of QM technology in the Air Force acquisition process. Near-term activities should enable initial use of QM technology. Long-term activities should enable the technology to mature.

2.4.1 Near-Term Implementation

Near-term activities should include trial programs to evaluate utility of the terminology, policy changes to initiate use, and education and training for familiarization.

Trial Programs. We recommend that QM technology be used on several trial programs prior to full implementation in the Air Force acquisition process. The purpose of the trial programs is to test acceptance and usefulness of QM technology and guidebooks. Programs selected should be representative of programs from each product division. These programs should be different than those selected to validate the methodology (see Sec. 5.0).

RADC should coordinate with the product divisions to identify candidate programs. The bases for selection should favor evolving the QM technology. RADC should work with computer resources focal points and software QA personnel at the product divisions to obtain data for selection.

Policy Changes. Policy change should include changes to regulations, standards, and DIDs that control using QM technology and to guidebooks addressing QA, reviews, and audits. Changes should be submitted early as it takes 1-3 months to incorporate changes to guidebooks and policies at the product division level and 6-12 months (or longer) to incorporate changes to policy at AFSC or Air Force level. While policy changes are being coordinated, a policy statement can be issued as an interim measure by higher-level headquarters to direct the use of quality metrics technology in software acquisitions.

Detailed recommendations for changes are described in Section 6.0. Recommended changes ensure that QM technology will be integrated into the Air Force acquisition

process and that use of QM technology will begin prior to specifying system-level requirements. We recommend that RADC contact the Office of Primary Responsibility (OPR) regarding each regulation, standard, and DID to be modified and coordinate change preparation.

Education and Training. We recommend an education and training program to prepare personnel in program offices for using QM technology because this technology is relatively new. The program should consist of two courses addressing different issues: software quality specification and software quality evaluation.

The specification course should be tailored to the needs of Air Force software acquisition managers, software engineers, and software QA personnel. This course should include topics such as an overview of software quality assurance and its evolution to date, benefits of using quality metrics technology, how to specify QM requirements, how to make QM trade-offs, how to use evaluation data to track progress, courses of action that should be taken based on evaluation information, and when data should be collected. The objective is to provide an understanding of the role of QM technology in software acquisition.

The evaluation course should be targeted to data collection and analysis personnel. Topics for this course should include when to collect data, quality framework elements, software products, data collection and analysis procedures, and automated data collection and analysis tools. The objective is to provide an understanding of quality factors and how to measure quality levels.

Both courses should be developed prior to using QM technology in the program offices. It is recommended that RADC determine the best way to develop these courses and initiate actions to enable development. RADC should also identify an OPR for training to be conducted.

2.4.2 Long-Term Implementation

Long-term activities should include selecting and validating metrics for each product division, establishing an historical data base, and automating portions of the procedural steps. Each product division should select metrics appropriate to their applications,

validate metrics, and establish factor ratings for use in different applications. A single, centralized data base should be established to enable validating metrics, developing factor ratings, and evaluating success of the application of QM technology. Portions of procedural steps should be automated to improve efficiency.

These activities place a requirement on the QM technology to be flexible for tailoring to applications and for accommodating changes resulting from validation efforts. QM foundation concepts (e.g., quality model and metric measurement) and the framework, in general, were shown to be sound and flexible through repeated application to a variety of programs. Many of the quality factors and metrics have been validated for general applications, and new and revised factors and metrics have been incorporated. Potential for growth and change is inherent in the QM technology.

Select and Validate Metrics. There are over 300 metric elements; some may not be appropriate for specific applications. Each product division should select a metric subset appropriate for its products that can then be tailored for specific applications. A product division may also elect to generate and validate new metrics and metric elements.

Establish Quality Metrics Database. When validating metrics and establishing factor ratings, data is required from different applications to perform correlations and comparisons. We recommend that the Data and Analysis Center for Software (DACS) at Rome be used as the data base for quality metrics information and that the SPO provide a copy of the quality requirements and all metric data to DACS (e.g., provide a copy of the Software Quality Evaluation Report). This has the advantages of providing one centralized location for all QM data and enabling access to all historical data by any one product division. It also enables large-scale data analysis and correlation to be performed on data from all product divisions. Any changes in QM technology such as new factors, metrics, and worksheet formats should be disseminated from a central point. This concept is illustrated in Figure 2.4-1.

Automatic Activities. Automating portions of procedural steps can improve efficiency of the process. Candidates for automation or automated support include trade studies considering factor interrelationships and life cycle costs, data collection and analyses, and report generation. When source information for collecting data is documentation,

Framework Elements:

- Factors
- Criteria
- Metrics
- Metric Elements
- Worksheets
- Scoresheets

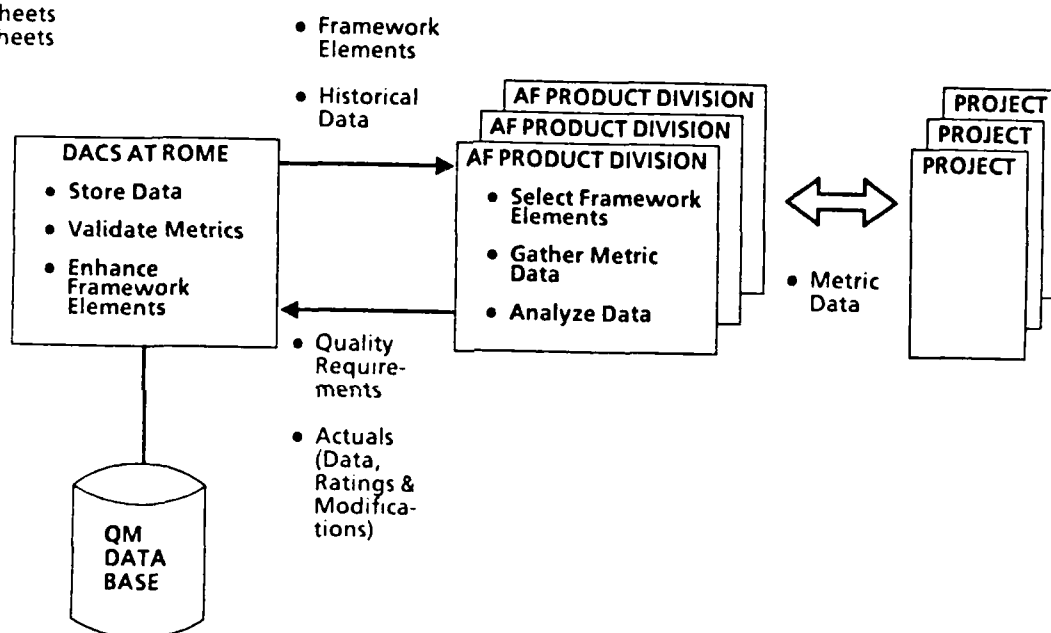


Figure 2.4-1 Relationship between Product Divisions and DACS

data normally entered on metric worksheets could be entered directly into a data base, and reasonableness checks could be performed on data entries. When source information is code, analyzers could be developed to gather metric data. The Automated Measurement Tool (AMT) has been developed to gather metric data from COBOL source code.

2.5 POTENTIAL BENEFITS AND PROBLEMS

This section discusses the potential benefits and problems associated with integrating QM technology into the software acquisition management process and of using QM technology during acquisition.

2.5.1 Benefits

Possible benefits of using QM technology include a higher quality end product, greater emphasis on quality throughout the life cycle, better management control, and life-cycle cost savings. A high-quality end product is possible because required quality levels are specified quantitatively. There is little room for misinterpretation or for undesirable results such as a highly efficient but unreliable and unmaintainable product. The acquisition manager is assured that the end product is of the required degree of quality. Also, other software requirements are considered at the same time that quality requirements are being specified. This means that the quality requirements should be reasonable and should not conflict with functional and performance requirements (or vice versa), thereby increasing the likelihood that all software requirements can be satisfied within allocated resources. In addition, achieved quality levels are monitored throughout development providing increased visibility for control of quality. Periodic application of metrics provides the acquisition manager with adequate feedback about software development progress and enables early redirection if necessary. Finally, evaluating specific low metric scores provides an additional mechanism for detecting deficiencies and anomalies in requirements, design, and code.

Life-cycle cost savings are possible for several reasons. Using metrics to detect deficiencies and anomalies enables correction during development. Correction at this time is less costly than during operation and maintenance. Also, it is possible to be

more precise about funding for quality. If adequate quality levels are achieved during development, it is unnecessary to spend more effort in raising quality levels or in developing a near-perfect product.

The greatest cost savings potential comes from having certain qualities actually built into the software. For example, if system A has a high level of reusability built into the software, then cost savings result from building system B reusing a portion of system A software. These potential cost savings are available for other quality factors such as flexibility, portability, interoperability, and expandability. Details for considering cost are described in Section 4.0 of the specification guidebook (Vol. II).

Other benefits can also be realized. For example, use of QM technology can provide the acquisition manager an added assurance that the required degree of reliability is achieved in the final product. This would be especially important in acquisitions involving space applications or nuclear armaments.

2.5.2 Problems

There are potential technical and administrative problems when using quality metrics in acquisitions; i.e., in integrating QM technology into the Air Force software acquisition process. Problems could arise during one of the most important tasks, that of maintaining a current QM technology baseline. Baseline changes could result from, for example, changes in quality factor ratings, new factor ratings being established, new metrics being established, and metrics being validated for new application areas. Changes could originate from any product division using QM technology. Using DACS would minimize the risk of such problems as: multiple baselines in the product divisions, duplication of validation efforts, and use of outdated information (e.g., outdated ratings).

A potential problem could arise where subjective judgment is required in scoring some metrics. Two people gathering metric data from the same software products could score the worksheets differently. This risk has been minimized by rewriting the questions on the metric worksheets so that they are clear, simple, and understandable. Also, metric element explanations have been included for clarification. As more historical information becomes available, it will be possible to do a reasonableness

check on worksheet data entries, based on previous data ranges. However, we recommend that experienced personnel perform data collection and that education and training be provided for personnel involved with QM technology.

Another potential problem might arise when attempting to automate portions of the data gathering task through an automated measuring tool. This type of tool scans source code and outputs statistics on the code (e.g., percentage of comments, number of specific constructs). The scanner is language dependent and must be developed for each language, but standardization on a language (e.g., Ada) will minimize cost.

Problems with organizational structures and manpower may be encountered when implementing QM technology at the product divisions. Program offices do not have QA divisions. QA in the program office is usually done by Engineering. In addition, software QA organizations in the product divisions are relatively new. These organizations are trying to define their role in the acquisition process and their relationship to the program offices. Absence of a well-defined organizational structure for software QA could lead to disagreements over assigning QM responsibilities. Either organization could resist accepting responsibility for QM functions because of staffing problems. Program offices are usually not fully staffed with software engineers; to accept more responsibilities without additional personnel would be difficult. Software QA organizations have small staffs and find it difficult to hire qualified personnel. A person with experience in both software engineering and QA is required, but few software engineers are interested in QA assignments. Staffing problems should receive attention during implementation of QM technology in the Air Force software acquisition process.

3.0 QUALITY METRICS FRAMEWORK

This section identifies enhancements made to the software quality framework in Task 2, Enhance Framework. The framework from the most recent Rome Air Development Center (RADC) quality measurement contract (described in RADC-TR-83-175) was used as the baseline. The following sections provide descriptions of elements of the enhanced framework: factors, criteria, metrics, metric worksheets, and factor scoresheets. Changes to the baseline framework are discussed after the description of each enhanced framework element.

An interim technical report (CDRL A003) for this contract was produced that describes an interim enhanced framework resulting from changes prior to July, 1983—approximately the midpoint of this contract. Pertinent baseline elements from RADC-TR-83-175 are also identified. Subsequent to the interim report, DOD-STD-SDS and the STARS DIDs became significant considerations in enhancing the framework. The following sections summarize the final enhanced framework and all changes resulting from this contract.

General Description. The goals of quality metrics (QM) technology are to enable a software acquisition manager to (1) specify the types and degrees of software qualities desired in the end product and (2) predict end-product quality levels through measuring the degree of those qualities present during development. The RADC quality program (see Sec. 1.0) has established a model for viewing software quality. Figure 1.3-1 depicts this model, showing a hierarchical relationship between a quality factor, criteria, and metrics. Criteria and metrics are factor attributes.

Quality factors (e.g., reliability, usability, correctness, and maintainability) are user-oriented terms, each representing an aspect of software quality. Thirteen quality factors are used to specify the types of qualities wanted in a particular software product. Product environment and expected use affect emphasis. For example, if human lives could be affected, integrity, reliability, correctness, verifiability, and survivability would be emphasized. If the software is expected to have a long life cycle, maintainability and expandability would be emphasized.

Criteria are software-oriented terms representing software characteristics. For example, operability and training are criteria for usability. The degree to which these characteristics are present in software is an indication of the degree of presence of an aspect of quality (i.e., a quality factor).

Metrics are software-oriented details of a characteristic (a criterion) of the software. Each metric is defined by a number of metric elements. The metric elements enable quantification of the degree of presence of criteria and, hence, factors. "Are all the errors specified which are to be reported to the operator/user?" is an example metric element question for the criterion operability (see worksheet 0, OP.1(2), App. A).

Using the methodology developed under this contract, the acquisition manager is responsible for specifying needed quality factors by priority, with quality levels commensurate with cost consideration. Factor requirements are provided as part of the software requirements (along with operational, performance, and design requirements). This enables the corresponding criteria and metrics to be identified and used to measure the degree of presence of desired qualities at key review points during development, allowing periodic predictions of the quality level for the final product. Metric worksheets and scoresheets help in applying the metrics and in determining metric scores.

Major Changes. Several major enhancements were made to the framework that simplify the framework conceptually and ease the tasks of specifying quality requirements and collecting metric data.

- a. Factors and criteria were grouped in three categories—performance, design, and adaptation—reflecting acquisition manager concerns.
- b. Metric worksheet information was recategorized to enable metric questions to be applied to specific products of phases described in DOD-STD-SDS.
- c. Metric questions were completely rewritten using terminology described in DOD-STD-SDS, using explanatory information and examples for clarity, and including formulas to correlate answers. Metric questions are nearly identical to questions in the Software Evaluation Reports proposed as part of the STARS measurement DIDs.

- d. New factor scoresheets were generated for translating worksheet information into scores for metric elements, metrics, criteria, and factors.

Rationale for change decisions include: (1) consistency within the framework, (2) ease of understanding and use by a software acquisition manager and by data collection and reduction personnel, and (3) compatibility with current DOD software technology thrusts.

3.1 SOFTWARE QUALITY FACTORS

Description. Thirteen software quality factors are identified in Table 2.2-1, with the user concern that characterizes the need for each type of quality. Quality factors are shown grouped under one of three acquisition concerns: performance, design, or adaptation. An acquisition manager specifying requirements for software will likely do so in a DOD-STD-SDS format in four main areas: (1) software performance characteristics (performance), (2) software design and construction (design), (3) anticipated software expansion or reuse (adaptation), and (4) quality assurance (including quality metrics). The similarity of areas and acquisition concerns enables the acquisition manager to easily identify and select quality factor categories and specific factors of interest. Quality criteria are similarly categorized (see Sec. 3.2); thus, selecting criteria and metrics is simplified.

Changes. The three acquisition concerns—performance, design, and adaptation—were formerly called life-cycle activities and were termed product operation, product revision, and product transition, respectively. The reason for this change is that it is easier for an acquisition manager to work with the new categories, as described in the prior paragraph.

Two factors were moved under a different category when category names were changed. Correctness, formerly under product operation (now performance), was placed under design (formerly product revision). Flexibility, formerly under product revision (now design), was placed under adaptation (formerly product transition). The reason for this change is that it enables criteria to be categorized under the same three acquisition concerns: performance, design, and adaptation. There were formerly no categories for criteria.

Table 3.1-1 Software Quality Factor Definitions and Rating Formulas

ACQUISITION CATEGORIES	QUALITY FACTOR	DEFINITION	RATING FORMULA
PERFORMANCE	EFFICIENCY	RELATIVE EXTENT TO WHICH A RESOURCE IS UTILIZED (e.g. STORAGE SPACE, PROCESSING TIME, COMMUNICATION TIME)	$1 - \frac{\text{ACTUAL RESOURCE UTILIZATION}}{\text{ALLOCATED RESOURCE UTILIZATION}}$
	INTEGRITY	EXTENT TO WHICH THE SOFTWARE WILL PERFORM WITHOUT FAILURES DUE TO UNAUTHORIZED ACCESS TO THE CODE OR DATA WITHIN A SPECIFIED TIME PERIOD	$1 - \frac{\text{ERRORS}}{\text{LINES OF CODE}}$
	RELIABILITY	EXTENT TO WHICH THE SOFTWARE WILL PERFORM WITHOUT ANY FAILURES WITHIN A SPECIFIED TIME PERIOD	$1 - \frac{\text{ERRORS}}{\text{LINES OF CODE}}$
	SURVIVABILITY	EXTENT TO WHICH THE SOFTWARE WILL PERFORM AND SUPPORT CRITICAL FUNCTIONS WITHOUT FAILURES WITHIN A SPECIFIED TIME PERIOD WHEN A PORTION OF THE SYSTEM IS INOPERABLE	$1 - \frac{\text{ERRORS}}{\text{LINES OF CODE}}$
	USABILITY	RELATIVE EFFORT FOR USING SOFTWARE (TRAINING AND OPERATION) (e.g. FAMILIARIZATION, INPUT PREPARATION, EXECUTION, OUTPUT INTERPRETATION)	$1 - \frac{\text{LABOR DAYS TO USE}}{\text{LABOR YEARS TO DEVELOP}}$
DESIGN	CORRECTNESS	EXTENT TO WHICH THE SOFTWARE CONFORMS TO ITS SPECIFICATIONS AND STANDARDS	$1 - \frac{\text{ERRORS}}{\text{LINES OF CODE}}$
	MAINTAINABILITY	EASE OF EFFORT FOR LOCATING AND FIXING A SOFTWARE FAILURE WITHIN A SPECIFIED TIME PERIOD	$1 - 0.1 (\text{AVERAGE LABOR DAYS TO FIX})$
	VERIFIABILITY	RELATIVE EFFORT TO VERIFY THE SPECIFIED SOFTWARE OPERATION AND PERFORMANCE	$1 - \frac{\text{EFFORT TO VERIFY}}{\text{EFFORT TO DEVELOP}}$
ADAPTATION	EXPANDABILITY	RELATIVE EFFORT TO INCREASE THE SOFTWARE CAPABILITY OR PERFORMANCE BY ENHANCING CURRENT FUNCTIONS OR BY ADDING NEW FUNCTIONS OR DATA	$1 - \frac{\text{EFFORT TO EXPAND}}{\text{EFFORT TO DEVELOP}}$
	FLEXIBILITY	EASE OF EFFORT FOR CHANGING THE SOFTWARE MISSIONS, FUNCTIONS, OR DATA TO SATISFY OTHER REQUIREMENTS	$1 - 0.05 (\text{AVERAGE LABOR DAYS TO CHANGE})$
	INTEROPERABILITY	RELATIVE EFFORT TO COUPLE THE SOFTWARE OF ONE SYSTEM TO THE SOFTWARE OF ANOTHER SYSTEM	$1 - \frac{\text{EFFORT TO COUPLE}}{\text{EFFORT TO DEVELOP}}$
	PORTABILITY	RELATIVE EFFORT TO TRANSPORT THE SOFTWARE FOR USE IN ANOTHER ENVIRONMENT (HARDWARE, CONFIGURATION AND/OR SOFTWARE SYSTEM ENVIRONMENT)	$1 - \frac{\text{EFFORT TO TRANSPORT}}{\text{EFFORT TO DEVELOP}}$
	REUSABILITY	RELATIVE EFFORT TO CONVERT A SOFTWARE COMPONENT FOR USE IN ANOTHER APPLICATION	$1 - \frac{\text{EFFORT TO CONVERT}}{\text{EFFORT TO DEVELOP}}$

NOTE: THE RATING VALUE RANGE IS FROM 0 TO 1. IF THE VALUE IS LESS THAN 0, THE RATING VALUE IS ASSIGNED TO 0.

The user concern for correctness was changed from a general concern for overall software performance to a specific concern for conformance to requirements (i.e., specifications and standards). The reason for this change is that attributes of correctness deal exclusively with format of software design and documentation; none deal with content material affecting software performance.

3.1.1 Factor Definitions and Rating Formulas

3.1.1.1 Description

Quality factor definitions and factor rating formulas are shown in Table 3.1-1. Rating formulas quantify user concerns for the final product. The formulas use three types of measurements: (1) number of errors per lines of code (2) effort to perform an action and (3) utilization of resources. Ratings should fall in the range from zero to one. The rating formula for reliability is one minus the number of errors per lines of code. For example, if one error per 1,000 lines of code occur during a given time period (e.g., during operational testing and evaluation) the rating formula shows a reliability level of 0.999($1 - 1/1,000 = 0.999$).

During software development, metrics are applied to software products, and a metric score is calculated for the appropriate factors. This metric score is an estimation (or prediction) of what the quality level will be for the final product. Figure 3.1-1 indicates the timeframes during which rating values are estimated through metric scores (closed box) and the timeframes during which rating values can be assessed by using actual data and the rating formula (dotted box). For example, the rating value for reliability is estimated by using metric scores during software development. During operational testing and evaluation and during production and deployment, actual data on number of errors per lines of code become available to assess the rating and evaluate predictions made during development. Exact correlations between metric scores and rating values have not been established. Research has only shown that higher metric scores during development result in higher quality end products. Table 3.1-2 shows a range of values for each rating formula that might occur when using actual data (e.g., during production and deployment) to assess rating values. The values shown are hypothetical.

The following paragraphs describe the factors and rating formulas in each acquisition concern category.

ACQUISITION CONCERN QUALITY FACTOR	APPLICATION PHASE	INITIAL USE OF PRODUCT			NEW USE OF PRODUCT		
		SOFTWARE DEVELOPMENT	OPERATIONAL TESTING AND EVALUATION	PRODUCTION AND DEPLOYMENT	SOFTWARE DEVELOPMENT	OPERATIONAL TESTING AND EVALUATION	PRODUCTION AND DEPLOYMENT
PERFORMANCE					<div> <div></div> <div>SAME AS FOR INITIAL USE AS REQUIRED</div> <div></div> </div>		
EFFICIENCY							
INTEGRITY							
RELIABILITY							
SURVIVABILITY							
USABILITY							
DESIGN					<div> <div></div> <div>SAME AS FOR INITIAL USE AS REQUIRED</div> <div></div> </div>		
CORRECTNESS							
MAINTAINABILITY							
VERIFIABILITY							
ADAPTATION							
EXPANDABILITY							
FLEXIBILITY							
INTEROPERABILITY			(AR)	(AR)			
PORTABILITY							
REUSABILITY							

= RATING ESTIMATION
 = RATING ASSESSMENT
 (AR) = AS REQUIRED

Figure 3.1-1 Rating Estimation and Rating Assessment Windows

Table 3.1-2 Quality Factor Ratings

Quality factor	Rating formula	Rating information			
Efficiency	1- $\frac{\text{Actual utilization}}{\text{Allocated utilization}}$	Value	0.1	0.3	0.5
		% utilization	90%	70%	50%
Integrity	1- $\frac{\text{Errors}}{\text{Lines of code}}$	Value	0.9995	0.9997	0.9999
		Errors/LOC	5/10,000	3/10,000	1/10,000
Reliability	1- $\frac{\text{Errors}}{\text{Lines of code}}$	Value	0.995	0.997	0.999
		Errors/LOC	5/1,000	3/1,000	1/1,000
Survivability	1- $\frac{\text{Errors}}{\text{Lines of code}}$	Value	0.9995	0.9997	0.9999
		Errors/LOC	5/10,000	3/10,000	1/10,000
Usability	1- $\frac{\text{Labor-days to use}}{\text{Labor-years to develop}}$	Value	0.5	0.7	0.9
		Days/years	5/10	6/20	10/100
Correctness	1- $\frac{\text{Errors}}{\text{Lines of code}}$	Value	0.9995	0.9997	0.9999
		Errors/LOC	5/10,000	3/10,000	1/10,000
Maintainability	1- 0.1 (average labor-days to fix)	Value	0.8	0.9	0.95
		Average labor-days	2.0	1.0	0.5
Verifiability	1- $\frac{\text{Effort to verify}}{\text{Effort to develop}}$	Value	0.4	0.5	0.6
		% effort	60%	50%	40%
Expandability	1- $\frac{\text{Effort to expand}}{\text{Effort to develop}}$	Value	0.8	0.9	0.95
		% effort	20%	10%	5%
Flexibility	1- 0.05 (average labor-days to change)	Value	0.8	0.9	0.95
		Average labor-days	40	20	10
Interoperability	1- $\frac{\text{Effort to couple}}{\text{Effort to develop}}$	Value	0.9	0.95	0.99
		% effort	10	5	1
Portability	1- $\frac{\text{Effort to transport}}{\text{Effort to develop}}$	Value	0.9	0.95	0.99
		% effort	10	5	1
Reusability	1- $\frac{\text{Effort to convert}}{\text{Effort to develop}}$	Value	0.4	0.6	0.8
		% effort	60	40	20

Performance. Performance quality factors deal both with the ability of the software to function and with error occurrences that affect software functioning. Low quality levels predict poor software performance. These quality factors are efficiency, integrity, reliability, survivability, and usability.

Efficiency deals with utilization of a resource. The rating formula for efficiency is in terms of actual utilization of a resource and budgeted allocation for utilization. For example, if a unit is budgeted for 10% available memory and actually uses 7%, the rating formula shows an efficiency level of 0.3 ($1 - 0.07/0.10 = 0.3$).

Integrity deals with software security failures due to unauthorized access. The rating formula for integrity is in terms of number of integrity-related software errors occurring during a given time (e.g., during operational testing and evaluation) and total number of executable lines of source code. This formula is similar to the formula for reliability; the difference is that reliability is concerned with all software errors, and integrity is concerned only with the subset of errors that affect integrity. For example, if three integrity-related errors per 10,000 lines of code occurred during operational testing and evaluation, the rating formula shows an integrity level of 0.9997 ($1 - 1/10,000 = 0.9997$).

Reliability concerns any software failure. The rating formula for reliability is in terms of total number of software errors occurring during a specified time and total number of executable lines of source code. For example, if three errors per 1,000 lines of code occurred during operational testing and evaluation, the rating formula shows a reliability level of 0.997 ($1 - 3/1,000 = 0.997$).

The concern with survivability is that software continue to perform (e.g., in a degraded mode) even when a portion of the system has failed. The rating formula for survivability is in terms of number of survivability-related errors (the subset of errors that affect survivability) occurring during a specified time and total number of executable lines of source code. This formula is similar to the formula for reliability.

Usability deals with relative effort involved in learning about and using software. The rating formula for usability is in terms of average effort to use software (to train for using it and to operate it) and original development effort. This formula considers size

of the software system in rating usability. It is recommended that effort to use be expressed in labor-days and effort for original development be expressed in labor-years to maintain a scoring range consistent with that of other factors. For example, if 10 labor-days were required for training on a system that required 100 labor-years to develop, the rating formula shows a usability level of 0.9 ($1 - 10/100 = 0.9$); and if five labor-days were required for training on a system that required 10 labor-years to develop, the rating formula shows a usability level of 0.5 ($1 - 5/10 = 0.5$).

Design. Design quality factors deal mainly with software failure and correction. Low quality levels usually result in repeating a portion of the development process (e.g., redesign, recode, reverify); hence the term design. The factors are correctness, maintainability, and verifiability.

Correctness deals with the extent to which software design and implementation conform to specifications and standards. Criteria of correctness (completeness, consistency, and traceability) deal exclusively with design and documentation formats. Under the three criteria there are no metrics dealing with content material affecting software operation or performance. The rating formula for correctness is in terms of number of specifications-related and standards-related errors that occur after formal release of the specifications and standards and total number of executable lines of source code. This formula is also similar to the formula for reliability; the difference is that correctness is concerned only with that subset of errors related to violations of specified requirements and nonconformance to standards.

Maintainability is concerned with ease of effort in locating and fixing software failures. The rating formula for maintainability is in terms of average number of labor-days to locate and fix an error within a specified time (e.g., during production and deployment). For example, if an average of 0.5 labor-days were required to locate and fix errors during production and deployment, the rating formula shows a maintainability level of 0.95 ($1 - (0.1 \times 0.5) = 0.95$).

Verifiability deals with software design characteristics affecting the effort to verify software operation and performance. The rating formula for verifiability is in terms of effort to verify software operation and performance and original development

effort. This formula is similar to the adaptation, effort-ratio formulas. For example, if 40% of the development effort is spent reviewing and testing software, the rating formula shows a verifiability level of 0.6 ($1 - 0.40/1.00 = 0.6$).

Adaptation. These quality factors deal mainly with using software beyond its original requirements, such as extending or expanding capabilities and adapting for use in another application or in a new environment. Low quality levels predict relatively high costs for new software use. Quality factors are expandability, flexibility, interoperability, portability, and reusability.

Expandability deals with relative effort in increasing software capabilities or performance. The rating formula for expandability is in terms of effort to increase software capability and performance and original development effort. For example, if five labor-months were spent enhancing software performance for software that originally took 100 labor-months to develop, the rating formula shows an expandability level of 0.95 ($1 - 5/100 = 0.95$).

Flexibility deals with ease of effort in changing software to accommodate changes in requirements. The rating formula for flexibility is in terms of average effort to change software to satisfy other (i.e., new or modified) requirements within a specified time. For example, if an average of one labor-day was required to modify software functioning during operational testing and evaluation, the rating formula shows a flexibility level of 0.95 ($1 - (0.05 \times 1) = 0.95$).

Interoperability is concerned with relative effort in coupling software of one system to software of one or more other systems. The rating formula for interoperability is in terms of effort to couple and original development effort and is similar to the formula for expandability.

Portability deals with relative effort involved in transporting software to another environment (e.g., different host processor, operating system, executive). The rating formula for portability is in terms of effort to transport software for use in another environment and original development effort and is similar to the formula for expandability.

Reusability is concerned with relative effort for converting a portion of software for use in another application. The rating formula for reusability is in terms of effort to convert software for use in another application and original development effort and is similar to the formula for expandability.

If adaptation effort is greater than original development effort, the effort-ratio formulas will yield a quality level value less than zero. In this case, the quality level value is assigned to zero. (This situation is considered unlikely because it would probably be less expensive to develop a new product than to adapt an existing one.)

3.1.1.2 Changes

Only eight of the 13 factors had rating formulas that were validated during prior contracts. Rating formulas were developed for the remaining five factors: efficiency, integrity, usability, correctness, and verifiability. Format for new formulas conforms to that for prior formulas (see Tbl. 3.1-1). The term "man-days" was changed to the neutral term "labor-days" for maintainability and flexibility.

Factor definitions were changed for correctness and flexibility. The definition of correctness was changed from "Extent to which the software satisfies its specifications and fulfills the user's mission objectives" to "Extent to which the software conforms to its specifications and standards". The reason is that attribute criteria (completeness, consistency, and traceability) deal exclusively with design and documentation formats. No attributes deal with content material affecting operation or performance (i.e., attributes do not deal with satisfying specifications or fulfilling user mission objectives). The definition of flexibility was changed from ". . .extending the software missions. . ." to ". . .changing the software missions. . ." because extension of capability is the domain of expandability.

The phrase "within a specified time period" was added to the definitions of integrity, reliability, survivability, and maintainability. Examples of time periods are during operational testing and evaluation and during the first year of operation and maintenance. Current literature emphasizes use of a definite time period for reliability modeling. Minor wording changes were made to factor definitions for the sake of consistency and clarity; intent of definitions was not changed.

Table 3.2-1 Software Quality Factors and Criteria

	ACQUISITION CONCERN			PERFORMANCE					DESIGN			ADAPTATION					
ACQUISITION CONCERN	FACTOR/ACRONYM			EFFICIENCY	INTEGRITY	RELIABILITY	SURVIVABILITY	USABILITY	CORRECTNESS	MAINTAINABILITY	VERIFIABILITY	EXPANDABILITY	FLEXIBILITY	INTEROPERABILITY	PORTABILITY	REUSABILITY	
	CRITERION/ACRONYM																E
PERFORMANCE	ACCURACY	AC				X											
	ANOMALY MANAGEMENT	AM			X												
	AUTONOMY	AU				X											
	DISTRIBUTEDNESS	DI				X											
	EFFECTIVENESS - COMMUNICATION	EC		X													
	EFFECTIVENESS - PROCESSING	EP		X													
	EFFECTIVENESS - STORAGE	ES		X													
	OPERABILITY	OP					X										
	RECONFIGURABILITY	RE				X											
	SYSTEM ACCESSIBILITY	SS			X												
TRAINING	TN						X										
DESIGN	COMPLETENESS	CP							X								
	CONSISTENCY	CS							X	X							
	TRACEABILITY	TC							X								
	VISIBILITY	VS								X	X						
ADAPTATION	APPLICATION INDEPENDENCE	AP															X
	AUGMENTABILITY	AT										X					
	COMMONALITY	CL												X			X
	DOCUMENT ACCESSIBILITY	DO													X		
	FUNCTIONAL OVERLAP	FO															X
	FUNCTIONAL SCOPE	FS															X
	GENERALITY	GE										X	X				X
	INDEPENDENCE	ID												X	X		X
	SYSTEM CLARITY	ST															X
	SYSTEM COMPATIBILITY	SY												X			X
VIRTUALITY	VR										X						
GENERAL	MODULARITY	MO				X				X	X	X	X	X	X	X	X
	SELF-DESCRIPTIVENESS	SD								X	X	X	X		X	X	X
	SIMPLICITY	SI			X					X	X	X	X				X

3.1.2 Quality Factor Interrelationships

Relationships exist among quality factors; some relationships are synergistic and others conflicting. Specifying requirements for more than one type of quality for a product can possibly have either a beneficial or an adverse effect on cost to provide the quality. Factor interrelationships are discussed in Section 4.0 because this aspect of the framework was enhanced when developing specification procedures during Task 3.

3.1.3 Quality Factor Relationships to Life-Cycle Phases

When using QM technology in acquiring a product, additional costs are associated with quality-related activities during software development. Benefits are also possible during software development and subsequent phases. Costs and benefits will vary for different factors, different factor combinations, and different activities within the life cycle. Relationships of quality factors to life-cycle phases are discussed in Section 4.0 because this aspect of the framework was enhanced when developing specification procedures during Task 3.

3.2 SOFTWARE QUALITY CRITERIA

3.2.1 Description

Criteria are software-oriented terms representing software characteristics. Software quality criteria can be grouped under the same three acquisition concerns as quality factors: performance, design, and adaptation. Table 3.2-1 shows the relationship of criteria to quality factors. Four categories for criteria are shown: performance, design, adaptation, and general. Each criterion is an attribute of one or more quality factors. The criteria in the first three categories are solely attributes of factors within the same acquisition concern (i.e., performance, design, and adaptation). Criteria in the fourth category are factor attributes within more than one acquisition concern.

Criteria and factors within each category are listed alphabetically for easy referencing. Alphabetizing by name or by acronym gives the same sequence. Criteria definitions are listed in Table 3.2-2.

Table 3.2-2 Quality Criteria Definitions

ACQ- UISI- TION			
	CRITERION	ACRONYM	DEFINITION
P E R F O R M A N C E	ACCURACY	AC	<ul style="list-style-type: none"> Those characteristics of software which provide the required precision in calculations and outputs
	ANOMALY MANAGEMENT	AM	<ul style="list-style-type: none"> Those characteristics of software which provide for continuity of operations under and recovery from non-nominal conditions
	AUTONOMY	AU	<ul style="list-style-type: none"> Those characteristics of software which determine its non-dependency on interfaces and functions
	DISTRIBUTEDNESS	DI	<ul style="list-style-type: none"> Those characteristics of software which determine the degree to which software functions are geographically or logically separated within the system
	EFFECTIVENESS-COMM	EC	<ul style="list-style-type: none"> Those characteristics of the software which provide for minimum utilization of communications resources in performing functions
	EFFECTIVENESS-PROCESSING	EP	<ul style="list-style-type: none"> Those characteristics of the software which provide for minimum utilization of processing resources in performing functions
	EFFECTIVENESS-STORAGE	ES	<ul style="list-style-type: none"> Those characteristics of the software which provide for minimum utilization of storage resources
	OPERABILITY	OP	<ul style="list-style-type: none"> Those characteristics of software which determine operations and procedures concerned with operation of software and which provide useful inputs and outputs which can be assimilated
	RECONFIGURABILITY	RE	<ul style="list-style-type: none"> Those characteristics of software which provide for continuity of system operation when one or more processors, storage units, or communication links fail
	SYSTEM ACCESSIBILITY	SS	<ul style="list-style-type: none"> Those characteristics of software which provide for control and audit of access to the software and data
D E S I G N	TRAINING	TN	<ul style="list-style-type: none"> Those characteristics of software which provide transition from current operation and provide initial familiarization
	COMPLETENESS	CP	<ul style="list-style-type: none"> Those characteristics of software which provide full implementation of the functions required
	CONSISTENCY	CS	<ul style="list-style-type: none"> Those characteristics of software which provide for uniform design and implementation techniques and notation
	TRACEABILITY	TC	<ul style="list-style-type: none"> Those characteristics of software which provide a thread of origin from the implementation to the requirements with respect to the specified development envelope and operational environment
V I S I B I L I T Y	VISIBILITY	VS	<ul style="list-style-type: none"> Those characteristics of software which provide status monitoring of the development and operation
	APPLICATION INDEPENDENCE	AP	<ul style="list-style-type: none"> Those characteristics of software which determine its nondependency on database system, microcode, computer architecture, and algorithms
	AUGMENTABILITY	AT	<ul style="list-style-type: none"> Those characteristics of software which provide for expansion of capability for functions and data
	COMMONALITY	CL	<ul style="list-style-type: none"> Those characteristics of software which provide for the use of interface standards for protocols, routines, and data representations
	DOCUMENT ACCESSIBILITY	DO	<ul style="list-style-type: none"> Those characteristics of software which provides for easy access to software and selective use of its components
	FUNCTIONAL OVERLAP	FO	<ul style="list-style-type: none"> Those characteristics of software which provide common functions to both systems
	FUNCTIONAL SCOPE	FS	<ul style="list-style-type: none"> Those characteristics of software which provide commonality of functions among applications
	GENERALITY	FE	<ul style="list-style-type: none"> Those characteristics of software which provide breadth to the functions performed with respect to the application
	INDEPENDENCE	ID	<ul style="list-style-type: none"> Those characteristics of software which determine its non-dependency on software environment (computing system, operating system, utilities, input, output routines, libraries)
	SYSTEM CLARITY	ST	<ul style="list-style-type: none"> Those characteristics of software which provide for clear description of program structure in a non-complex and understandable manner
A D A P T A T I O N	SYSTEM COMPATIBILITY	SY	<ul style="list-style-type: none"> Those characteristics of software which provide the hardware, software, and communication compatibility of two systems
	VIRTUALITY	VR	<ul style="list-style-type: none"> Those characteristics of software which present a system that does not require user knowledge of the physical, logical, or topological characteristics
	MODULARITY	MO	<ul style="list-style-type: none"> Those characteristics of software which provide a structure of highly cohesive components with optimum coupling
	SELF-DESCRIPTIVENESS	SD	<ul style="list-style-type: none"> Those characteristics of software which provide explanation of the implementation of functions
G E N E R A L	SIMPLICITY	SI	<ul style="list-style-type: none"> Those characteristics of software which provide for definition and implementation of functions in the most noncomplex and understandable manner

3.2.2 Changes

Criteria were categorized under the same three acquisition concerns as quality factors: performance, design, and adaptation. This was possible because of the recategorization of quality factors. There were formerly no categories for criteria. This change simplifies selection of factor attributes.

Both criteria and factors were organized alphabetically within each acquisition concern. Some criteria acronyms were changed so that alphabetizing by name or by acronym gives the same sequence for easy referencing.

The number of criteria was changed from 30 to 29. Specificity and conciseness were deleted; metrics for these criteria (formerly SP.1 and CO.1) were placed under simplicity (as SI.5 and SI.6, respectively) with metrics that are similar in scope. Communicativeness was deleted; metrics for this criteria (formerly CM.1 and CM.2) were placed under operability (as OP.2 and OP.3, respectively) because both deal with operational usability. Effectiveness was expanded to three criteria: effectiveness-communication, effectiveness-processing, and effectiveness-storage. Metrics for effectiveness were placed under the new criteria. The new criteria were created to enable differentiation of concerns for communication, processing, and storage efficiency at the criteria, rather than metric, level.

Minor wording changes were made to criteria definitions for the sake of consistency and clarity. Intent of the definitions was not changed.

3.3 SOFTWARE QUALITY METRICS

3.3.1 Description

Metrics are software-oriented details of a software characteristic (a criterion). Each criterion consists of one or more metrics. Each metric is an attribute of only one criterion. Table 3.3-1 lists the name and acronym of each criterion (in alphabetical order) and the name and acronym of each metric that is an attribute of that criterion. Metric acronyms are acronym extensions of the parent criterion. For example, the acronym for the criterion commonality is CL; the acronym for the three metric attributes are CL.1, CL.2, and CL.3.

Table 3.3-1 Quality Metrics Summary

CRITERION		METRIC	
NAME	ACRONYM	NAME	ACRONYM
ACCURACY	AC	ACCURACY CHECKLIST	AC.1
ANOMALY MANAGEMENT	AM	ERROR TOLERANCE/CONTROL IMPROPER INPUT DATA COMPUTATIONAL FAILURES HARDWARE FAULTS DEVICE ERRORS COMMUNICATIONS ERRORS NODE/COMMUNICATION FAILURES	AM.1 AM.2 AM.3 AM.4 AM.5 AM.6 AM.7
APPLICATION INDEPENDENCE	AP	DATA BASE MANAGEMENT IMPLEMENTATION INDEPENDENCE DATA STRUCTURE ARCHITECTURE STANDARDIZATION MICROCODE INDEPENDENCE FUNCTIONAL INDEPENDENCE	AP.1 AP.2 AP.3 AP.4 AP.5
AUGMENTABILITY	AT	DATA STORAGE EXPANSION COMPUTATION EXTENSIBILITY CHANNEL EXTENSIBILITY DESIGN EXTENSIBILITY	AT.1 AT.2 AT.3 AT.4
AUTONOMY	AU	INTERFACE COMPLEXITY SELF-SUFFICIENCY	AU.1 AU.2
COMMONALITY	CL	COMMUNICATIONS COMMONALITY DATA COMMONALITY COMMON VOCABULARY	CL.1 CL.2 CL.3
COMPLETENESS	CP	COMPLETENESS CHECKLIST	CP.1
CONSISTENCY	CS	PROCEDURE CONSISTENCY DATA CONSISTENCY	CS.1 CS.2
DISTRIBUTEDNESS	DI	DESIGN STRUCTURE	DI.1
DOCUMENT ACCESSIBILITY	DO	ACCESS TO DOCUMENTATION WELL-STRUCTURED DOCUMENTATION	DO.1 DO.2
EFFECTIVENESS-COMMUNICATION	EC	COMMUNICATION EFFECTIVENESS MEASURE	EC.1
EFFECTIVENESS-PROCESSING	EP	PROCESSING EFFECTIVENESS MEASURE DATA USAGE EFFECTIVENESS MEASURE	EP.1 EP.2
EFFECTIVENESS-STORAGE	ES	STORAGE EFFECTIVENESS MEASURE	ES.1
FUNCTIONAL OVERLAP	FO	FUNCTIONAL OVERLAP CHECKLIST	FO.1
FUNCTIONAL SCOPE	FS	FUNCTION SPECIFICITY FUNCTION COMMONALITY FUNCTION SELECTIVE USABILITY	FS.1 FS.2 FS.3
GENERALITY	GE	UNIT REFERENCING UNIT IMPLEMENTATION	GE.1 GE.2
INDEPENDENCE	ID	SOFTWARE INDEPENDENCE FROM SYSTEM MACHINE INDEPENDENCE	ID.1 ID.2
MODULARITY	MO	MODULAR IMPLEMENTATION MODULAR DESIGN	MO.1 MO.2
OPERABILITY	OP	OPERABILITY CHECKLIST USER INPUT COMMUNICATIVENESS USER OUTPUT COMMUNICATIVENESS	OP.1 OP.2 OP.3
RECONFIGURABILITY	RE	RESTRUCTURE CHECKLIST	RE.1
SELF-DESCRIPTIVENESS	SD	QUANTITY OF COMMENTS EFFECTIVENESS OF COMMENTS DESCRIPTIVENESS OF LANGUAGE	SD.1 SD.2 SD.3
SIMPLICITY	SI	DESIGN STRUCTURE STRUCTURED LANGUAGE OR PREPROCESSOR DATA AND CONTROL FLOW COMPLEXITY CODING SIMPLICITY SPECIFICITY HALSTEAD'S LEVEL OF DIFFICULTY MEASURE	SI.1 SI.2 SI.3 SI.4 SI.5 SI.6

Table 3.3-1 Quality Metrics Summary (continued)

CRITERION		METRIC	
NAME	ACRONYM	NAME	ACRONYM
SYSTEM ACCESSIBILITY	SS	ACCESS CONTROL ACCESS AUDIT	SS.1 SS.2
SYSTEM CLARITY	ST	INTERFACE COMPLEXITY PROGRAM FLOW COMPLEXITY APPLICATION FUNCTIONAL COMPLEXITY COMMUNICATION COMPLEXITY STRUCTURE CLARITY	ST.1 ST.2 ST.3 ST.4 ST.5
SYSTEM COMPATIBILITY	SY	COMMUNICATION COMPATIBILITY DATA COMPATIBILITY HARDWARE COMPATIBILITY SOFTWARE COMPATIBILITY DOCUMENTATION FOR OTHER SYSTEM	SY.1 SY.2 SY.3 SY.4 SY.5
TRACEABILITY	TC	CROSS REFERENCE	TC.1
TRAINING	TN	TRAINING CHECKLIST	TN.1
VIRTUALITY	VR	SYSTEM/DATA INDEPENDENCE	VR.1
VISIBILITY	VS	UNIT TESTING INTEGRATION TESTING CSCI TESTING	VS.1 VS.2 VS.3

Each metric is defined by one or more metric elements. Metric elements are detailed questions applied to software products; answers to them enable quantification of metrics and of the parent criterion and factor. Metric elements are designated by acronym only (no name) and are listed on the metric worksheets. Acronym designation is an extension of the parent metric acronym. For example, the 14 metric element acronyms for the metric CL.1 are CL.1 (1) through CL.1 (14).

3.3.2 Changes

Several metrics were changed in conjunction with criteria changes, as mentioned in Section 3.2.2. The metric of specificity, SP.1, was changed to simplicity metric SI.5. The conciseness metric, CO.1, was changed to simplicity metric SI.6. The two communicativeness metrics, CM.1 and CM.2, were changed to operability metrics OP.2 and OP.3, respectively. The four effectiveness metrics were placed under three new criteria. EF.1 was placed under effectiveness-communication as EC.1. EF.2 and EF.3 were placed under effectiveness-processing as EP.1 and EP.2, respectively. EF.4 was placed under effectiveness-storage as ES.1.

Minor changes were made to metric names to help clarify the characteristic being measured and to be consistent in naming style. The terms "checklist" and "measure" were dropped from most metric names because they are misleading; metric elements are most often a combination of checklist and measurement questions. The term was retained when the name resulting from dropping the term might be confused with a criterion name.

Extensive changes were made to wording of metric elements for consistency, clarity, and ease of understanding. However, the software characteristic being measured by the metric, as the collection of metric elements, was generally not changed. The purpose of this enhancement was to enable a better understanding of metric elements by users. Validity of metrics was not questioned and no new metrics were added. Many examples were added to metric descriptions to help clarify intent; and explanatory information, formerly contained in a separate appendix, was integrated into the text. Terminology was changed to be consistent with proposed DOD-STD-SDS; for example, module was changed to unit. Changes were also made to the worksheets; some format changes affected metric elements and are described in Section 3.4.

Two changes were made at the metric level. Function completeness, formerly FS.3, was integrated with FS.2, function commonality, as they overlapped in scope. Function selective usability was changed from DO.3 to FS.3 because this metric deals with functional scope. The following paragraphs highlight changes at the metric element level that affect scope of the parent metric. Changes are identified by metric acronym and name.

AM.7 Node/Communication Failures. The scope of this metric was expanded from interoperating network nodes to include any interoperating system.

AP.2 Data Structure. The intent of central control of data could not be determined; so this aspect of the metric was dropped.

AP.3 Architecture Standardization. A standard, 32-bit computer architecture was unclear and considered inappropriate; so this aspect of the metric was dropped.

AP.5 Functional Independence. One intent of this metric is to determine general applicability of algorithms (i.e., not unique to one application). A table-driven algorithm was changed from the only way to achieve functional independence to an example of one method. The need for comments with algorithm code was considered the domain of self-descriptiveness and moved to SD.2, effectiveness of comments.

AT.4 Design Extensibility. Performance/price information for enhancement trades is not typically required in software DIDs; so this aspect of the metric was dropped.

AU.2 Self-sufficiency. The aspect of this metric dealing with word-processing capability was dropped because it deals with development environment rather than the operational software product.

CL.1 Communications Commonality. Uniform message handling was added as a related aspect of network communication.

CP.1 Completeness Checklist. A new element was added to determine whether all defined data items are referenced.

CS.1 Procedure Consistency. A new element was added to determine whether all references to the same function use a single, unique name.

CS.2 Data Consistency. A new element was added to determine whether all references to the same data use a single, unique name.

DI.1 Design Structure. The meaning of logical structure and function being separated in the design was unclear; so this aspect of the metric was dropped.

EC.1 Communication Effectiveness Measure. This metric currently has one element dealing with specification of performance requirements and limitations. The element was considered generally applicable to efficiency and was added to metrics EP.1, EP.2, and ES.1.

EP.1 Processing Effectiveness Measure. An aspect of this metric deals with internal communication time between software elements (e.g., units). This is not typically measured and may require special tools; so this aspect was dropped. A new element was added dealing with specification of performance requirements and limitations.

EP.2 Data Usage Effectiveness Measure. A new element was added dealing with specification of performance requirements and limitations.

ES.1 Storage Effectiveness Measure. The element of this metric dealing with virtual storage was deleted because it is redundant with an aspect of virtuality, VR.1. A new element was added dealing with specification of performance requirements and limitations.

SD.2 Effectiveness of Comments. The need for comments with algorithm code was moved to this metric from AP.5, functional independence.

ST.1 Interface Complexity. The element of this metric dealing with interface nesting levels was deleted because the meaning was unclear.

VS.2 Integration Testing. The element dealing with testing of specified performance requirements was placed under the metric dealing with performance testing: VS.3, CSCI testing.

VS.3 CSCI Testing. The element dealing with testing of specified performance requirements was moved to this metric from VS.2, integration testing.

3.4 METRIC WORKSHEETS

3.4.1 Description

Metric worksheets are contained in Appendix A. The worksheets contain metric elements as questions. Software products (specifications, documents, and source listings) are used as source information to answer questions on worksheets; answers are then translated into metric element scores (yes = 1, no = 0, and a formula answer results in a score from 0 to 1). This enables scoring of the parent metric, criterion, and factor and results in a quality level indication for the product.

Seven different worksheets are applied in different development phases. Table 3.4-1 indicates the timeframe during an acquisition life-cycle phase when a worksheet is used, shows the software level of abstraction at which the worksheet is applied, and lists key terminology used within the worksheet.

Worksheet 0 is applied to products of system/software requirements analysis. The worksheet is applied at the system level. (For large systems, software may not be a discernible component in the design with separate requirements at the system level. In this case, worksheet 0 is applied at the system segment level.)

Worksheet 1 is applied to products of software requirements analysis. A separate worksheet is used for each CSCI.

Worksheet 2 is applied to products of preliminary design. A separate worksheet is used for each CSCI.

Worksheets 3A and 3B are applied to products of detailed design. A separate worksheet 3A is used for each CSCI. A separate worksheet 3B is used for each unit of a CSCI. Worksheets 3A and 3B are applied together; answers on 3B worksheets for CSCI units are used in scoring the 3A worksheet for that CSCI.

Table 3.4-1 Metric Worksheet/Life-Cycle Correlation

Life-Cycle Phase/ Activity		Demonstration & Validation		Full-Scale Development (FSD)						
		System/ Software Requirements Analysis	Software Requirements Analysis	Preliminary Design	Detailed Design	Coding & Unit Testing	CSC Integration & Testing	CSCI - Level Testing	System Integration & Testing	
Application Level/ Terminology										
System	<ul style="list-style-type: none">• System• System function• CSCI	Metric Worksheet 0					<div>(Selected metric questions are reapplied during the integration and testing phases as indicated in the quality attribute correlation table in Appendix A.)</div>			
CSCI	<ul style="list-style-type: none">• CSCI• Software function	Metric Worksheet 1								
CSCI	<ul style="list-style-type: none">• CSCI• Top-level CSC	Metric Worksheet 2								
CSCI	<ul style="list-style-type: none">• CSCI• Top-level CSC• Lower-level CSC• Unit	Metric Worksheet 3A		Metric Worksheet 4A						
UNIT	<ul style="list-style-type: none">• Unit	Metric Worksheet 3B		Metric Worksheet 4B						

Worksheets 4A and 4B are applied to products of code and unit testing. Worksheets 4A and 4B are applied in the same manner as 3A and 3B. A separate worksheet 4A is used for each CSCI, and a separate worksheet 4B is used for each CSCI unit.

For the remainder of the development cycle, selected metric questions are reapplied as indicated in the quality attribute correlation table in Appendix A.

Metric worksheets are designed to be applied to software development products identified in DOD-STD-SDS. The minimum product set is listed by software development phase in Table 3.4-2. Each product is identified by title and by DID number. Information from the entire set of products for a particular phase is needed as source material to answer metric questions on the worksheet applicable to that phase. It is not necessary to specify the complete product set for each acquisition, only to have equivalent information available to answer worksheet questions. For example, when acquiring a small system, information regarding the QA plan and software standards may be included as part of the software development plan.

3.4.2 Changes

Metric worksheets and metric element questions on worksheets were revised extensively. Changes were incorporated to enable worksheets to be applied during phases defined in DOD-STD-SDS to products identified in DOD-STD-SDS. Terminology was also revised to be consistent with that used in DOD-STD-SDS (e.g., unit rather than module and CSCI rather than CPCI). Formulas for relating metric element data had formerly been contained in a separate appendix; these were integrated into the appropriate questions on the worksheets. Explanatory material for clarifying the intent of worksheet questions had formerly been contained in a separate appendix; pertinent explanatory material was integrated into the text of questions, and examples were included where appropriate. The end result is standalone worksheets compatible with DOD-STD-SDS.

The baseline framework assumed five general software development phases: requirements analysis, preliminary design, detailed design, implementation, and test and integration. Product descriptions for these phases were general. Products for phases described in DOD-STD-SDS are described in detail by DIDs (see Tbl. 3.4-2).

Table 3.4-2 Software Development Products

Phase/Product Title	Applicable DID
System/Software Requirements Analysis	
System/Segment Specification	DI-S-X101
Software Development Plan	DI-A-X103
Preliminary Software Requirements Specification	DI-E-X107
Operational Concept Document	DI-M-X125
Software Quality Assurance Plan	DI-R-X105
Software Problem/Change Report	DI-E-X106
Software Standards and Procedures Manual	DI-M-X109
Preliminary Interface Requirements Specification	DI-E-X108
Software Requirements Analysis	
Software Requirements Specification	DI-E-X107
Interface Requirements Specification	DI-E-X108
Preliminary Design	
Software Top-Level Design Document	DI-E-X110
Software Test Plan	DI-T-X116
Preliminary Software User's Manual	DI-M-X121
Preliminary Computer System Operator's Manual	DI-M-X120
Detailed Design	
Software Detailed Design Document	DI-E-X111
Software Test Description	DI-T-X117
Data Base Design Document	DI-E-X113
Interface Design Document	DI-E-X112
Coding and Unit Testing	
Source Code/Listings	(Appendix)
Preliminary Software Test Procedure	DI-T-X118
CSC Integration and Testing	
Software Test Procedure	DI-T-X118
CSCI-Level Testing	
Software Product Specification	DI-E-X114
Software Test Report(s)	DI-T-X119
Software User's Manual	DI-M-X121
Computer System Operator's Manual	DI-M-X120
System Integration and Testing	
Software Product Specification	DI-E-X114
Software Test Report(s)	DI-T-X119
Software User's Manual	DI-M-X121
Computer System Operator's Manual	DI-M-X120

Metric questions were recategorized among worksheets so that questions can be answered using specific DIDs as source material. This resulted in many changes in applicability of metric elements to worksheets and to specific wording of questions. The end result is worksheets that can be applied to development products identified in DOD-STD-SDS.

One problem with the baseline worksheets is that CSCI-level and CSCI component-level questions were mixed on several worksheets. Separation of question for application to software components or to software configuration items was left to the user. Creation of worksheets 3B and 4B for application to CSCI units solves this problem and simplifies scoring using worksheet data.

We worked closely with Dynamics Research Corporation (DRC) in revising metric questions to be compatible with terminology, phases, and products for DOD-STD-SDS. DRC had been responsible for drafting proposed DOD-STD-SDS and was drafting a set of DIDs for the STARS measurement task. A subset of those DIDs was to address software quality measurement and was to be compatible with Boeing Aerospace Company (BAC) efforts for the current RADC quality measurement contract (this contract). DRC produced nine Software Evaluation Report DIDs. BAC produced the seven worksheets (see Tbl. 3.4-1 and App. A). Wording of questions on the DIDs and worksheets is nearly identical. There are distinct differences between formats for the DIDs and worksheets because they differ in purpose.

The worksheets and eight of the DIDs are designed to be applied to products of the eight life-cycle activities identified in Table 3.4-1. The ninth DID is designed to be applied during system performance testing. Separate worksheets are provided for system/software requirements analysis through coding and unit testing; questions from these worksheets are reapplied during subsequent test and integration activities. A separate DID is provided for each of nine life-cycle activities; metric questions in the DIDs for test and integration activities are a restatement of questions from prior DIDs. (In other words, there is only a format difference between worksheets and DIDs in this respect.) The worksheets include formulas for relating metric element data. The DIDs collect only raw data and contain no formulas. The worksheets identify each question by metric element acronym. The DIDs identify parent criterion name and acronym and metric name and acronym and list questions sequentially under each

metric (i.e., no metric element acronyms). CSCI-level questions and unit-level questions are separated on different worksheets (worksheets 3A, 3B and 4A, 4B). Each DID contains all questions for one phase and identifies the application level for each question. Several minor differences occur in metric names (e.g., use of checklist or measure in the name).

There are no differences between the worksheets and DIDs in intent or scope of any metric. However, there are differences as to when metric questions are applied. The primary difference is that, although questions in worksheets and DIDs correspond during early development phases, sometimes a worksheet question is reapplied during a test and integration phase and this question does not appear on the corresponding test and integration DID. The primary focus of the joint effort with DRC was rewording questions. Schedule constraints on DID delivery prohibited comparing and revising task outputs. Differences in reapplication of metric questions were retained because of the concern for updating worksheet information that may change during test and integration activities.

3.5 FACTOR SCORESHEETS

3.5.1 Description

Factor scoresheets are contained in Appendix B. There are 13 factor scoresheets, one for each software quality factor. Scoresheets are used for translating information at the metric element level on the worksheets into a quality level score for a quality factor. Each scoresheet has blanks for the factor and for all attributes of that factor (i.e., criteria, metrics, and metric elements). Worksheet information is transferred to the scoresheets at the metric element level. "Yes" answers are scored as 1; "no" answers are scored as 0; and numeric answers resulting from formulas are transferred directly to scoresheets (scoring range from 0 to 1). Scores are then calculated for the parent metrics, criteria, and factor according to the hierarchical (attribute) relationship indicated on the scoresheet.

3.5.2 Changes

The baseline used metric tables for scoring metric elements and metrics. Scoring of criteria and factors was left to the user. Factor scoresheets enable all scoring to be done on one form for each factor: metric elements, metrics, criteria, and factor.

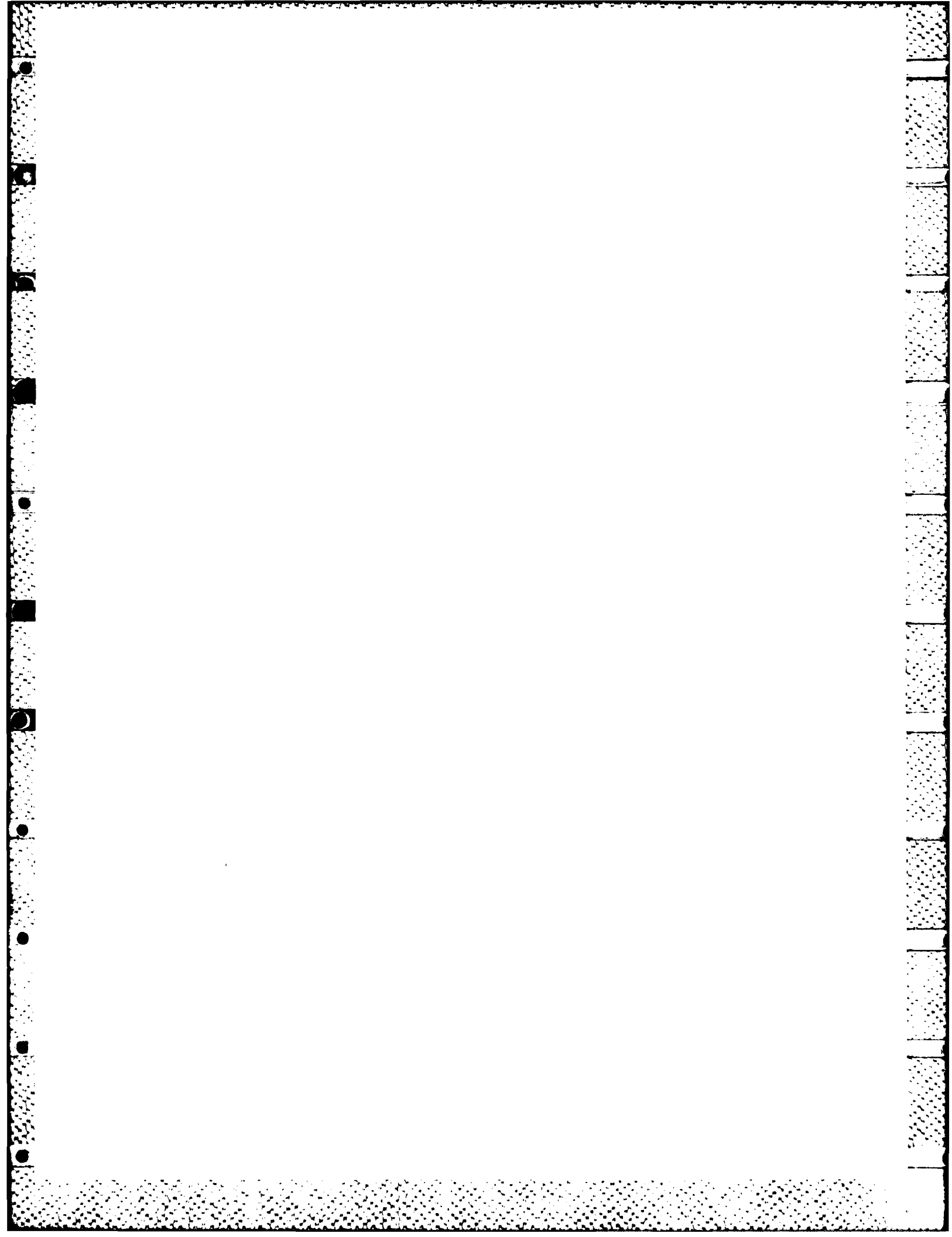
3.6 REFERENCES

The framework elements from the most recent RADC quality measurement contract (F30602-80-C-0330) were used as the baseline for enhancements. Framework elements are described in two volumes.

- a. RADC-TR-83-175 (Vol. I), "Software Quality Measurement for Distributed Systems-Final Report", July 1983.
- b. RADC-TR-83-175 (Vol. II), "Guidebook for Software Quality Measurement", July 1983.

Current literature, reports, and results of related contracts were examined when enhancing the framework to identify potential problem areas and suggestions for change. The following contract reports were useful.

- a. Contract F30602-80-C-0265, RADC-TR-83-174 (Vol. I), "Final Report - Software Interoperability and Reusability", July 1983.
- b. Contract F19628-80-C-0207: (1) ESD-TR-82-143 (Vol. I), "Introduction and General Instructions for Computer Systems Acquisition Metrics Handbook", May 1982; and (2) ESD-TR-82-143 (Vol. II), "Quality Factor Modules-Computer Systems Acquisition Metrics", May 1982.
- c. Contract DAAK80-79-D-0035, MD-81-RAPS-002, "Radar Prediction System Metric Evaluation", November 1981.



4.0 QUALITY METRICS METHODOLOGY

This section summarizes results of Task 3, Develop Methodology. The methodology from the most recent RADC quality measurement contract (described in RADC-TR-83-175) was used as a baseline. This baseline was extensively revised and enhanced. The focus of enhancing the methodology was to provide an acquisition manager a means for determining and specifying quality factor requirements for command and control applications. Emphases were placed on (1) techniques for choosing appropriate factors and quality-level requirements through considering quantitative assessments of factor interrelationships and factor life-cycle costs, (2) procedures for specifying requirements, and (3) procedures for analyses of quality measurement data.

The methodology, procedures, and techniques are documented in Volume II, Software Quality Specification Guidebook. Because enhancements to the quality framework and methodology were extensive, a second guidebook was developed to address the needs of data collection and analysis personnel. Procedures for data collection, analysis, and reporting are contained in Volume III, Software Quality Evaluation Guidebook. A specification of format and content for a report describing results of evaluation of software quality was also developed. The specification is in data item description (DID) format in Appendix C.

4.1 OVERVIEW

The following paragraphs provide an overview of the quality metrics (QM) methodology. Subsequent sections highlight features of the methodology and procedures developed for this contract.

Figure 4.0-1 shows the QM methodology in two major parts: software quality specification and software quality evaluation. Specification is the responsibility of the software acquisition manager and includes specifying software quality requirements and assessing compliance with those requirements. Results of the compliance assessment are used to initiate corrective action. The specification guidebook, Volume II, provides procedural guidance. Evaluation is the responsibility of data

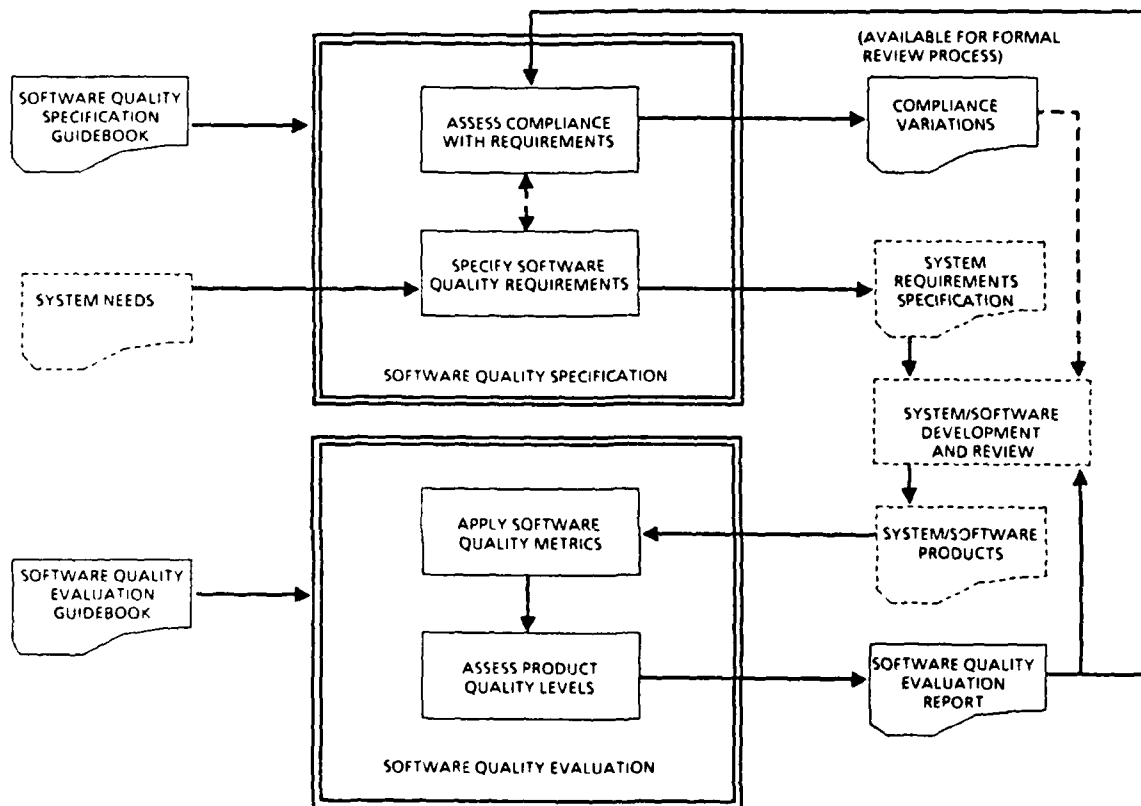


Figure 4.0-1 Software Quality Specification and Evaluation Process

collection and analysis personnel and includes applying software quality metrics to products of the development cycle, scoring product quality levels, and reporting results. The evaluation guidebook, Volume III, provides procedural guidance.

The process begins early in the system life cycle—usually during system demonstration and validation. We assume that a description of the nature of the system and system needs or requirements exists. This description could be a statement of work or a draft system specification and is the primary basis for identifying software quality factor requirements. A series of procedural steps is performed to determine specific software quality needs and to specify quality requirements. Steps include polling groups such as the Air Force using command and the Air Force Logistics Command (AFLC) in order to provide a comprehensive set of operational and support quality requirements from a quality factor point of view. These steps could be performed by the SPO or the development contractor or through awarding a separate contract.

Software quality requirements are entered into the system requirements specification and are treated as contractual obligations (just the same as technical requirements). As the system contractor proceeds with development, quality requirements from the system requirements specification are allocated to lower level specifications and finally assigned to units within the software detailed design document in a manner similar to that for other requirements. This requirements flow is shown in Figure 4.0-2. Each time during the cycle that a development product is released (usually at major review points such as system design review (SDR), software specification review (SSR), preliminary design review (PDR), and critical design review (CDR)), quality metrics, in the form of metric worksheets, are applied to the products. Raw data are then used to calculate scores indicating quality level achieved for each quality factor, and these scores are compared to specified requirements.

Application of metrics and scoring of achieved product quality levels are performed by the development contractor to show compliance with quality requirements. It is anticipated that product evaluation will also be performed in parallel by another group such as an IV&V team, the AFPRO, SPO Software Engineering, or Product Division Software Quality Assurance, as is discussed in Section 2.3. Data collection and analysis results are documented in a Software Quality Evaluation Report (see App. C). This report is reviewed separately at major review points. The report is included in

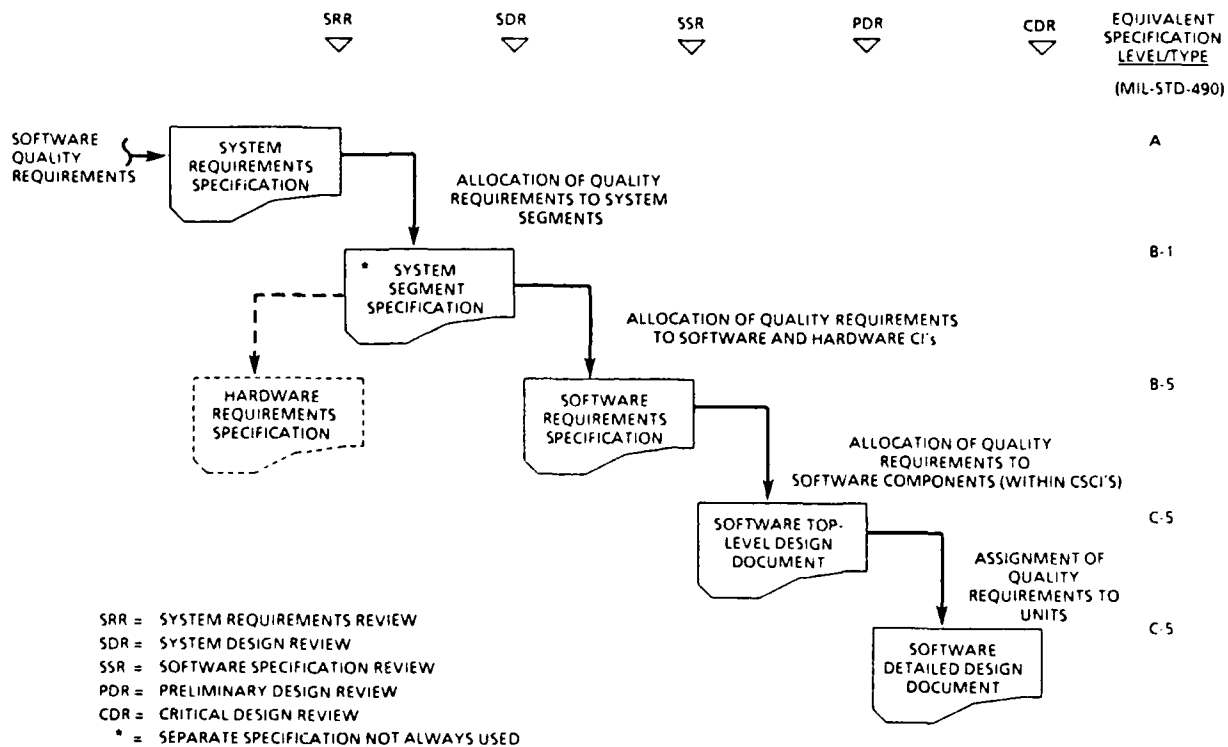


Figure 4.0-2 Flow of Software Quality Requirements

the review package released before the review date. The SPO uses these results to assess compliance with quality requirements and (1) approves or disapproves of compliance variations at the review and/or (2) respecifies quality requirements and ensures that changes are reflected in the system requirements specification.

Advantages of this methodology include:

- a. Quality factor requirements are determined concurrently with technical performance and design requirements—increasing the likelihood of a comprehensive set of requirements which can be satisfied within allocated schedule and budget.
- b. Requirements are allocated in a manner similar to the technical performance and design requirements—allocations can be checked from the system-level downward, and all lower-level requirements should be traceable.
- c. Progress on achieved quality levels is assessed periodically—enabling corrective action to be taken in a timely manner.
- d. Reporting of data collection and analysis results is comprehensive—providing adequate information for decision-making on a course of corrective action.

4.2 FEATURES

This section highlights features of the methodology and procedures developed or enhanced during this contract to support determining and specifying quality requirements. For a complete description of procedural steps, refer to Volume II, specification guidebook.

System-Level Focus. Specification of requirements is focused at the system rather than software (CSCI) level. Quality requirements for software are derived by examining a system description during system/software requirements analysis, so that requirements reflect system and user needs.

Software quality factor requirements are specified for each system level function that is supported by software. This enables freedom in designing software to optimally meet the needs of all applicable system-level functions, yet ensures that all system-level quality requirements are satisfied by the software design. This approach parallels the approach to specifying and allocating technical performance and design

requirements and increases the probability of obtaining a complementary set of technical and quality requirements that are realistic. This approach also avoids developing high quality levels for parts of the software not requiring high quality; this situation can occur when requirements are specified for software as a whole, rather than functionally.

Procedural Levels. Procedures are organized by quality level-of-detail. Separate procedures are provided for specifying requirements for factors, criteria, and metrics. This approach parallels attribute levels in the hierarchical software quality model and simplifies procedures.

Quality Goals. Three categories of importance (quality goal levels) are used when specifying quality factor requirements: excellent (E), good (G), and average (A). The categories enable differentiating among quality level concerns for performing factor trade studies and avoid the complications of using a numeric goal level.

We recommend using a numeric range when stating factor goal levels in specifications, after trade studies have been completed. Separate ranges can be specified for each goal level. Different ranges can be specified for different applications and for different functions within the same application.

Factor Dependencies and Interrelationships. Choosing quality factors and initial goal levels is primarily a matter of determining and translating system and user needs. Selecting achievable goal levels for the combination of factors chosen can be a complicated process because of relationships among factors. We developed detailed procedural steps to aid the acquisition manager in developing a realistic set of quality goals. Positive and negative factor interrelationships are quantified to indicate the degree of affect among factors.

Cost Considerations. Selecting achievable goal levels also involves considering costs. Developing high quality levels most often requires additional budget during the early phases of full-scale development (FSD) and most often results in cost savings (or cost avoidance) during production and deployment. We developed a detailed analysis of quality-related activities potentially affecting software life-cycle costs for each factor. Procedures aid the acquisition manager in determining relative cost variations

AD-A153 988

SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES VOLUME 1 2/2

FINAL REPORT(U) BOEING AEROSPACE CO SEATTLE WA

T P BOWEN ET AL. FEB 85 D182-11678-1

UNCLASSIFIED

RADC-TR-85-37-VOL-1 F30602-82-C-0137

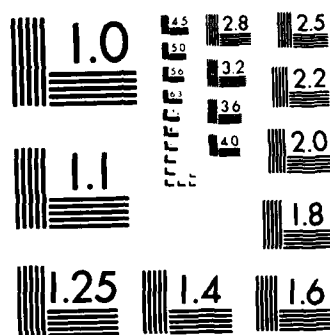
F/G 9/2

NL

END

FILED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

for the set of quality factors chosen and for refining quality goal levels based on cost effects of positive and negative factor relationships.

Criteria Weighting. Criterion attributes of a factor may require different emphasis, depending on the application and set of factors chosen. Weighting formulas are used to show the specific relationship between a factor and its attribute criteria. Each criterion is assigned a weighting value to indicate its percentage contribution to the overall factor goal. This approach communicates the desired emphasis on software characteristics for development and states the formula required for scoring.

We developed detailed procedures to aid the acquisition manager in determining the appropriate emphasis on each attribute criterion. Procedures include considerations of the application and effects of factor interrelationships at the criterion level.

Software Quality Evaluation Report. One important aspect of the QM methodology is ensuring that quality requirements have been satisfied. The iterative nature of the methodology ensures that the acquisitions manager has ample opportunity to take corrective action in a timely manner. The Software Quality Evaluation Report provides the manager with adequate information to assess compliance with requirements and to decide an appropriate course for any corrective action.

This report is contained in Appendix C in DID format. It requires that raw measurement data, scoring data, scoring trends, analyses information, and corrective recommendations be provided for all software quality factor requirements.

Command and Control Example. An example is used throughout the specification procedures for a command and control application. Procedural steps are applied to the example system and results are explained. A large-scale command and control system was used, and personnel familiar with system-level and software-level requirements and design were consulted. This approach aided in developing realistic procedures and provides the acquisition manager with an added depth of information.

5.0 VALIDATION PLAN

This section summarizes results of Task 4, Develop Validation Plan. The purpose of this task was to prepare a plan for validating the methodology and procedures for determining and specifying software quality requirements. The methodology and procedures were developed in Task 3 and are documented in the specification guidebook (Vol. II). The following paragraphs describe an approach for evaluating the usability of the specification methodology and procedures within the software acquisition management process.

The goal of the methodology is to enable an acquisition manager to acquire a software product which satisfies user quality needs. There are two major parts to the process: specifying software quality requirements and evaluating achieved software quality levels. Quality requirements are specified early in the software development process and are included with performance and development requirements. Achieved quality levels are evaluated at key review points in the development process such as PDR and CDR. Success of the methodology depends not only on determining and specifying realistic quality requirements and obtaining objective evaluations of achieved quality levels but also on responding to scoring that indicates a variation from specified quality requirements. This may involve redirecting development efforts and reevaluating specified quality requirements in order to acquire a final product which satisfies quality goals.

Validation efforts should focus on clarity, applicability, and effectiveness of the methodology and support material (specification guidebook, procedures, figures, tables, and Software Quality Evaluation Report). The methodology should be applied during the development phases of at least two projects, concurrently. Projects should be medium-sized command and control applications. (Application to large projects may involve extraneous complexity; application to small projects may not validate applicability.) Project development phases and review points should be compatible with those defined in DOD-STD-SDS. Information required by system and software deliverables should be compatible with information required by DIDs listed in Table 3.4-2.

A rating scheme should be developed to evaluate the methodology and support materials and consider the following, at a minimum:

- a. Usability of the methodology within the acquisition management process and soundness of approach.
- b. Relative ease of learning, understanding, and applying procedures.
- c. Clarity and completeness of procedures.
- d. Clarity and completeness of software quality factors and criteria.
- e. Appropriateness of metrics selected for use.
- f. Reasonableness of specified software factor goals and criteria weighting.
- g. Completeness and clarity of Software Quality Evaluation Report.
- h. Completeness of compliance assessment.
- i. Effectiveness of actions taken for compliance variations.

Questionnaires should be used to support the rating process.

Records should be maintained and should include:

- a. Problems encountered with the methodology and its implementation.
- b. "Side effect" problems resulting from introducing the methodology into an existing process (e.g., conflict of technical requirements and quality requirements).
- c. Costs and schedules associated with implementing the methodology, by phase.
- d. Risks identified with implementing the methodology.
- e. Unexpected benefits of the methodology.
- f. Qualifications of personnel associated with the effort.

Showing success of this methodology depends, in part, on correlating specified quality with the achieved quality; and this depends, in part, on validity of the quality metrics used. Since not all metrics have been validated, evaluations of achieved quality should be supported by independent assessments (e.g., questionnaires) and factor rating calculations whenever possible. This will minimize the possibility of variations in results due to the metrics themselves rather than the methodology.

6.0 RECOMMENDED REVISIONS

This section describes recommended Air Force documentation updates resulting from Task 6, Recommend Revisions. Content and wording changes to documentation used by an Air Force System Program Office (SPO) are recommended in order to implement quality metrics (QM) technology in the software acquisition process. Section 2.0 describes the role of QM technology in software acquisition; recommended revisions in the following sections ensure that pertinent aspects of QM technology are identified as specific required elements in software acquisition.

6.1 REVIEW AND RECOMMENDATION PROCESS

Documentation considered for review includes regulations, directives, specifications, standards, guidebooks, and data item descriptions (DID). The purpose of these documents in the acquisition process is to establish common directions and goals for both government and contractor personnel. There are two basic document categories: internal and compliance.

Regulations, directives, and guidebooks are internal documents and only apply to government personnel. These documents define roles and responsibilities of government personnel during the acquisition process and identify what should be required of a contractor. These documents are not binding on a contractor. Standards, specifications, and DIDs are compliance documents and impose requirements on a contractor. They direct what the contractor must do during the acquisition process. Change recommendations in the following sections are influenced by the scope of effect of each document.

The first step was to compile an initial list of internal and compliance documents potentially impacted by the implementation of QM technology (see Tbl. 6.1-1). Guidebooks and proposed Department of Defense (DOD) standards were later added to the list. Guidebooks were added because they recommend practices influenced by QM technology. Proposed DOD software standards were added because of the planned influence on software acquisition. Two proposed standards were added: DOD-STD-SDS and DOD-STD-SQS (formerly DOD-STD-SQAM). DOD-STD-SDS is now available

Table 6.1-1 Candidate Documents

TYPE		NUMBER	TITLE
INTERNAL	DOD Document	DOD 4120 3-M DODD 5000 1 DODD 5000 2 DODD 5000 3 DODD 5000 19 DOD 5000 19-L, Volume II DODD 5000 29 DODD 5000 31 DODD 5010 12 DODD 5010 19 DOD 7935 1-5	Standardization of Policies, Procedures, and Instructions Major System Aquisitions Major System Aquisition Process Test and Evaluation Policies for the Management and Control of DOD Information Requirements Acquisition Management Systems and Data Requirements Control List (AMSDL) Management of Company Resources in Major Defense Systems Interim List of DOD Approved High Order Programming Languages (HOL) Management of Technical Data Configuration Management Automated Data Systems Documentation Standards
	AF Regulation	AFR 65-3 AFR 73-1 AFR 74-1 AFR 80-14 AFR 300-2 AFR 800-2 (CI) AFR 800-3 AFR 800-4 AFR 800-11 AFR 800-14, Volume I AFR 800-14, Volume II AFR 800-19	Configuration Management Defense Standardization Program Quality Assurance Program Research and Development Test and Evaluation Management of the USAF Automatic Data Processing Program Acquisition Management - Program Management Engineering for Defense Systems Transfer of Program Management Responsibility Life Cycle Costing (LCC) Management of Computer Resources in Systems Acquisition and Support Procedures for Computer Resources in Systems System or Equipment Turnover
COMPLIANCE	Military Specification	MIL-Q-9858A MIL-S-52779 MIL-S-83490	Quality Program Requirements Software Quality Assurance Program Requirements Specifications, Types and Forms
	Military Standard	MIL-STD-130E MIL-STD-480 MIL-STD-481A MIL-STD-482A MIL-STD-483(USAF) MIL-STD-490 MIL-STD-499A MIL-STD-881A MIL-STD-1521A(USAF) MIL-STD-1588(USAF) MIL-STD-1589(USAF) MIL-STD-1679(Navy) MIL-STD-1015	Identification Marking of U.S. military Property Configuration Control - Engineering Changes, Deviations and Waivers Configuration Control - Engineering Changes, Deviations and Waivers (Short Form) Configuration Status Accounting Data Elements and Related Features Configuration Management Practices for Systems, Equipment, Munitions and Computer Programs Specification Practices Engineering Management Work Breakdown Structures for Defense Material Items Technical Reviews and Audits for Systems, Equipments, and Computer Programs JOVIAL (J3) JOVIAL (J73/1) Tactical Software Development Reference Manual for the Ada Programming Language

in draft form and is due to be released soon. Release of this standard and its associated DIDs will impact other software acquisition documents; therefore, these standards are the primary focus of review and detailed change recommendations. DOD-STD-SQS was released in draft form as DOD-STD-SQAM, but, due to industry review comments, it is scheduled to be rewritten. Only general change recommendations are included for this standard.

The next step was to review the candidate list of documents to determine which ones affect software quality during the acquisition process. The list was narrowed for a more detailed review where recommended changes were formulated. The final result was the recommended revisions in the following sections.

6.2 REVIEW ANALYSIS

We anticipate that QM technology will be applied only to selected contracts during the next few years as part of a validation process and that recommended changes to policy formally implementing QM technology will not be made immediately. Therefore, other policy changes likely to occur during this period should be taken into account.

If changes were to be made today, the primary focus would be on AFR 800-14, MIL-STD-490, MIL-STD-483, MIL-S-52779, and MIL-STD-1521A. But the two proposed DOD software standards change this focus. DOD-STD-SDS is expected to be released in January, 1985 and DOD-STD-SQS possibly one year later. Together, they should include all software development requirements to be imposed on contractors. The following paragraphs discuss affects of these two DOD standards on each of the documents named above.

AFR 800-14 is the primary regulation directing Air Force personnel in software acquisition. Changes to this regulation are required to direct Air Force personnel to implement QM technology. This regulation is already in an update process due to other policy changes and will undergo even further changes due to DOD-STD-SDS. It may be two years before this regulation is updated and released. In the interim, Air Force System Command (AFSC) and Air Force Logistics Command (AFLC) will issue a joint letter directing the use of DOD-STD-SDS in lieu of policy in AFR 800-14. Changes caused by DOD-STD-SDS will be significant and will affect those areas that

would require change to implement QM technology. Therefore, specific changes to AFR 800-14 are not recommended at this time. General recommendations are made in the next section that should be considered when updating AFR 800-14.

MIL-STD-490 includes information about type (A, B, C, D, and E) specifications. Of particular interest are sections on the B-5, Computer Program Development Specification, and C-5, Computer Program Product Specification. These sections and references to them are being modified to be consistent with DOD-STD-SDS. The type B-5 will become software development specification and will include the software requirements specification and the interface requirements specification. The type C-5 will become software product specification and will include the software top-level design document, software detailed design document, data base design document, and interface design document. References will be made to the DIDs in DOD-STD-SDS for content and format descriptions. Because of these modifications, no changes will be required to MIL-STD-490 to implement QM technology.

MIL-STD-483 presently contains appendices describing CPCI specifications. Changes are being made to make MIL-STD-483 consistent with DOD-STD-SDS. Several appendices are being deleted (concerning software specifications) because content information is either inconsistent or redundant with information contained in DOD-STD-SDS and its associated DIDs. Because of these modifications, no further changes will be required in order to implement QM technology.

MIL-S-52779 is expected to be replaced by DOD-STD-SQS and will not be imposed on contractors in the future. Therefore, changes to MIL-S-52779 are unnecessary.

MIL-STD-1521A addresses technical reviews and audits. The proposed update (MIL-STD-1521B) has been modified to incorporate two new reviews and additional information pertaining to other reviews. New sections have been added for the software specification review (SSR) and test readiness review (TRR) required by DOD-STD-SDS. Major modifications have been made to sections covering the system design review (SDR), preliminary design review (PDR) and critical design review (CDR) to accommodate the evolution of information required by DOD-STD-SDS. QM technology recommends reviewing certain data at these reviews. Thus, some changes to the updated MIL-STD-1521B are required.

6.3 DETAILED RECOMMENDED CHANGES

DOD-STD-SDS and DOD-STD-SQS and their associated DIDs are the primary focus for recommended changes to implement QM technology. General recommendations are included for other documents. The list of documents recommended for change is shown in Table 6.3-1.

6.3.1 DOD-STD-SDS

Quantitative quality factor requirements should be specified at the highest development level by the program office. The contractor's responsibility is to ensure that the requirements are flowed down to the CSCI level as appropriate and to define a methodology that will be used to demonstrate compliance with requirements. Specification of requirements will be ensured by DOD-STD-SDS, while the definition of a methodology to show compliance will be required by DOD-STD-SQS. The following changes are recommended to DOD-STD-SDS.

- a. Section 5.1: Rewrite the first sentence as follows, "The contractor shall establish a complete set of functional, performance, interface and quantitative quality requirements for each CSCI."
- b. Section 5.1.1.4: Add after item (4), "(5) the specification of quantitative quality requirements," and change item (5) to item (6).
- c. Section 5.1.3: Change "and interface requirements" in the second sentence to read, "interface, and quality requirements . . ."
- d. Section 5.6: Add the following section, "Section 5.6.1.4 The contractor shall demonstrate compliance with the quality requirements in accordance with the Software Quality Assurance Plan." It is recognized that the software quality assurance plan may be renamed the software quality program plan or software quality evaluation plan. This recommended change should be modified appropriately.
- e. Section 5.8: In item (2), insert before "evaluation" the word "quantitative".

Table 6.3-1 Documentation Recommended for Revision

TYPE	NUMBER	TITLE
REGULATION	AFR 800-14, Vol II	Acquisition and Support Procedures for Computer Resources in Systems and AFSC SUPPLEMENT to AFR 800-14
STANDARD	MIL-STD-1521A/B DOD-STD-SDS DOD-STD-SQS	Technical Reviews and Audits for Systems, Equipment, and Computer Programs Defense System Software Development (Draft) Software Quality Standard (Draft)
DID	DI-S-X101 DI-R-X105 DI-E-X107	System/Segment Specification Software Quality Assurance Plan Software Requirement Specification
GUIDEBOOK	ASD-TR-78-7 ASD-TR-77-(?) ESD-TR-78-117 ASD-TR-78-8 ASD-TR-78-47 ESD-TR-77-255	Airborne Systems Software Acquisition Engineering Guidebooks: Reviews and Audits Software Acquisition Engineering Guidebook Series: Reviews and Audits Software Acquisition Management Guidebooks: Reviews and Audits Airborne Systems Software Acquisition Engineering Guidebooks: Quality Assurance Software Acquisition Engineering Guidebook Series: Software Quality Assurance Software Acquisition Management Guidebooks: Software Quality Assurance

- f. Section 5.8.2: Insert the word "quantitative" prior to "system" in the first and second sentences.
- g. Section 5.8.2a: Rewrite to read "That quantitative quality requirements for factors such as efficiency, integrity, reliability, survivability, usability, correctness, maintainability, verifiability, flexibility, portability, expandability, reusability, and interoperability have been established."

6.3.2 DI-S-X101 System/Segment Specification

Section 3.4.3: Replace with the following, "Additional Quality Factors. This paragraph shall specify additional quality factor requirements, not mentioned in prior paragraphs, in quantitative terms and any specific conditions under which the requirements are to be met."

6.3.3 DI-R-X105 Software Quality Assurance Plan

This DID will be rewritten as the software quality program plan or software quality evaluation plan. Specific changes cannot be recommended at this time. Either way, a change should be made to require the contractor to define the methodology and framework that will be used to demonstrate compliance with the quantitative quality requirements. A specific QM methodology should not be imposed on the contractor. Therefore, the contractor should be required to define that methodology which will be used in this document.

6.3.4 DI-E-X107 Software Requirements Specification

This DID is presently written to encourage the specification of quality factors in quantitative terms. If the final version does not differ from this version in Section 3.7, only a minor change will be required as follows:

Section 3.7.7: Change "Testability" to "Verifiability".

6.3.5 DOD-STD-SQS

This proposed DOD software standard will require the contractor to establish a software quality program. According to recent decisions, DOD-STD-SDS will cover that part of the software quality program that affects software development, while DOD-STD-SQS will cover evaluation activities. The next draft of DOD-STD-SQS is due out for review in January, 1985.

Certain changes should be made to DOD-STD-SQS to implement QM technology. A requirement should be added to identify the specific metrics and framework that the contractor will use to show compliance with the quality requirements for software. A proposed data item description (DID) has been developed as part of this contract to report results of evaluations at various points in time. This DID, Software Quality Evaluation Report (see App. C), should be approved to be used with DOD-STD-SQS. The standard should also be modified to discuss use of this DID throughout the life cycle and the specific information to be evaluated at each review point. Quality factors should be updated to be consistent with the list in this report. These changes should be consistent with DOD-STD-SDS so that the two DOD software standards complement each other.

6.3.6 MIL-STD-1521B

This military standard addresses technical reviews and audits conducted during the software development process. It is expected that MIL-STD-1521B will be released with only minor changes to its current draft form. Thus, the following recommended changes are to the proposed revision B, rather than MIL-STD-1521A.

- a. Section 10.3.1.4 (SRR): Add item, "d. The quality factors to be included in the requirements and the quantitative values for them."
- b. Section 20.1 (SDR): Add "and quality" after the word "test" in the first sentence.
- c. Section 20.3.1 (SDR): Change item g to read, "Hardware and software quality requirements."

- d. Section 20.3 (SDR): Add item "20.3.14 Review the software quality evaluation report to ensure that applicable quality requirements are being satisfied."
- e. Section 30.2 (SSR): In item g, change "testability" to "verifiability, survivability, expandability."
- f. Section 30.2 (SSR): Add item "h. The software quality evaluation report for evaluation against the quality requirements."
- g. Section 40.2.2 (PDR): Add item "n. Review the software quality evaluation report for evaluation against the quality requirements."
- h. Section 50.2.2 (CDR): Add item "e. The software quality evaluation report for evaluation against the quality requirements."
- i. Section 70.4.12 (FCA): Add item "g. The software quality evaluation report shall be reviewed to ensure that all quality requirements for software have been met."

6.3.7 AFR 800-14

As was stated earlier, this regulation will be changed extensively to be made consistent with DOD-STD-SDS. When QM technology is implemented, the following general changes are recommended to the revised AFR 800-14.

Volume I covers general policy on management of computer resources. An item should be added to Section 3 directing that a trade-off study be conducted on quality factors for software so that quality requirements can be specified at the system level in quantitative terms based on the framework and methodology to be implemented.

Volume II defines detailed procedures to be used in the software acquisition process by Air Force personnel. Changes should eventually be made to the information contained in the present Sections 2-3, 2-4, and 2-8 to generally describe the additional activities in the various system acquisition and software development phases that are a part of using QM technology. This includes the specification of quality factors (and related trade studies) and the quantitative evaluation of quality levels at various points in the acquisition life cycle.

In terms of planning, the information in present Sections 3-2, 3-4, 3-6, and 3-7 should be modified to identify the specification of quality factors for the software in quantitative terms and to describe trade studies conducted in order to determine requirements for quality factors. It is assumed that Section 3-9 (Computer Program Development Plan) will be removed or revised to be consistent with DOD-STD-SDS. The last changes should be made to information presently in Section 4-9 (Formal Technical Reviews) to incorporate appropriate changes to be consistent with those recommended to MIL-STD-1521 in Section 6.3.6 of this report.

6.3.8 Guidebooks

Changes to guidebooks are not required to implement QM Technology as they are not policy documents. They are used to teach new Air Force personnel about policies. As such, it would be useful, though, to update certain guidebooks when implementing QM technology. It is assumed that these guidebooks will be updated in the near future to reflect changes resulting from the implementation of DOD-STD-SDS and DOD-STD-SQS. These modifications will be extensive for the guidebooks addressing quality assurance.

Three of the guidebooks address reviews and audits. These three guidebooks will require minor updates to incorporate information reflecting changes in MIL-STD-1521 resulting from implementation of QM technology. Specific information to be added to these guidebooks include a discussion of quantitative requirements for quality factors and use of the Software Quality Evaluation Report at each of the various formal reviews as appropriate (SDR, SSR, PDR, CDR, and FCA). The three revised guidebooks should be evaluated for appropriate changes when QM technology is implemented.

The other three guidebooks address software quality assurance and are based heavily on AFR 800-14 and MIL-S-52779. With the revisions to AFR 800-14 and the replacement of MIL-S-52779 due to DOD-STD-SDS and DOD-STD-SQS, these guidebooks will probably be completely rewritten. These guidebooks will then discuss an entire software quality program consistent with the two new DOD software standards. Some general information should be incorporated into these guidebooks about QM technology along with references to the two new guidebooks developed under this

contract (see Vol. II and Vol. III): Software Quality Specification Guidebook and Software Quality Evaluation Guidebook.

General subjects that should be covered in summary form include (1) a discussion about using QM technology in the acquisition process, (2) responsibilities for specification and evaluation of quality factors, (3) initial planning to include a software quality metric framework and trade studies necessary to determine requirements, (4) use of the Software Quality Evaluation Report at various review points, (5) an overall orientation of how QM technology fits into each life-cycle phase, (6) how it is to be applied to subcontractors, (7) identification of quality factors and a definition for each, and (8) the relationship between existing guidebooks and the two new guidebooks developed under this contract. Detailed directions concerning which quality factors to include, how to specify quantitative requirements for them, and how to evaluate them using QM technology are included in the new guidebooks. These two guidebooks will complement the existing ones.

APPENDIX A

METRIC WORKSHEETS

(The contents of this appendix are in Vol. III., App. A)

APPENDIX B

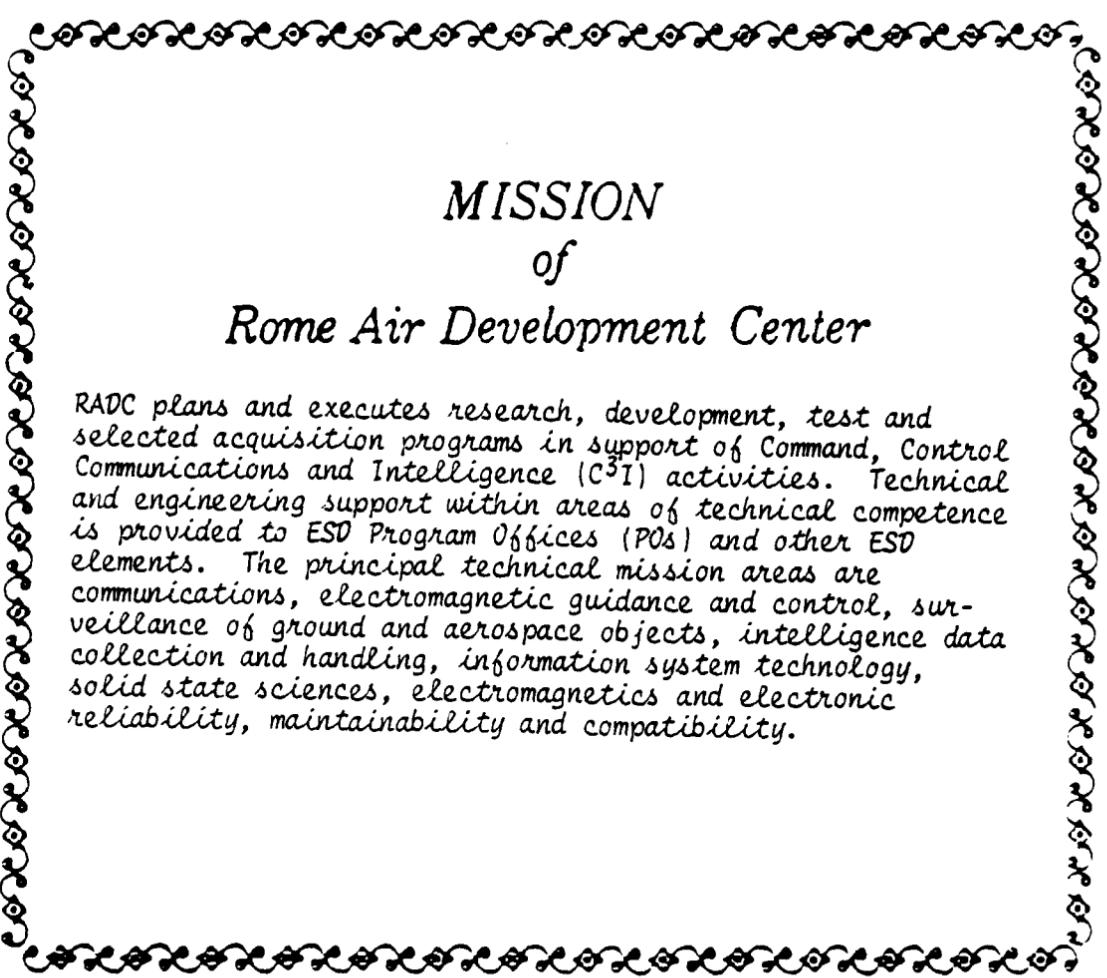
FACTOR SCORESHEETS

(The contents of this appendix are in Vol. III, App. B.)

APPENDIX C

SOFTWARE QUALITY EVALUATION REPORT

(The contents of this appendix are in Vol. II, App. C and Vol. III, App. C.)



*MISSION
of
Rome Air Development Center*

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C³I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, solid state sciences, electromagnetics and electronic reliability, maintainability and compatibility.

END

FILMED

6-85

DTIC